

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

Новое в жизни, науке, технике

Подписная научно-популярная серия

Издается ежемесячно с 1988 г.

В океане данных

Системы управления базами данных

1988

10

Новое
в жизни,
науке,
технике

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

Подписная
научно-
популярная
серия

10/1988

Издается
ежемесячно
с 1988 г.

В ОКЕАНЕ ДАННЫХ

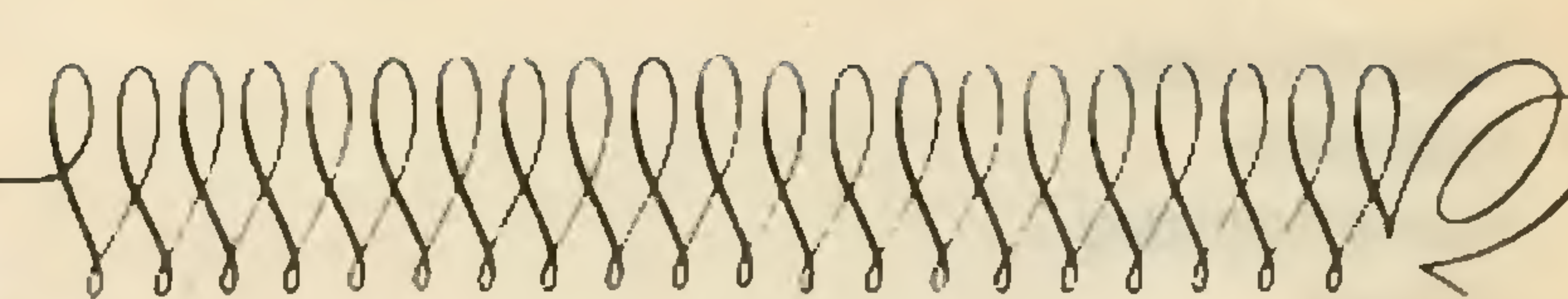




Издательство
«Знание»
Москва
1988

ББК 22.19
В93





БЕЛОВ Виктор Александрович — заведующий сектором Всесоюзного научно-исследовательского института проблем вычислительной техники и информатики. Кандидат технических наук. Круг научных интересов лежит в области систем управления базами данных и знаний и программного обеспечения персональных компьютеров. Имеет около 40 научных работ и два изобретения.

МАЛЫХИНА Мария Петровна — кандидат технических наук, доцент, программист.

ЧАСТИКОВ Аркадий Петрович — кандидат технических наук, доцент, специализируется в области информатики и вычислительной техники. Имеет около 80 опубликованных работ, среди них научно-популярные, отмеченные поощрительными дипломами.

ФРОЛОВ Степан Алексеевич — кандидат технических наук, начальник отдела Всесоюзного межотраслевого центра «Интеграл».

ВАДЯСКИН Юрий Ефимович — начальник сектора Всесоюзного межотраслевого центра «Интеграл».

АЛЕКСАКОВ Габриэль Николаевич — кандидат технических наук, доцент МИФИ. Имеет более 60 печатных работ, в том числе более 30 изобретений. В 1964 г. разработал первый советский транзисторный телевизор «Малахит».

РЕДАКТОР Б. М. ВАСИЛЬЕВ



Мало кто задумывается о том, что базы данных БД и системы управления базами данных (СУБД) придуманы для того, чтобы облегчить человеку жизнь, а отнюдь не для того, чтобы ее усложнить. Мало кто знает, что современная техника, в частности персональные компьютеры, и современные СУБД позволяют пользоваться базами данных даже тем, кто абсолютно не знаком с программированием. От этой неосведомленности теряет в конце концов все общество, так как главная задача автоматизированных информационных систем — экономить самый ценный общественный ресурс — время.

В ОКЕАНЕ ДАННЫХ

В. А. БЕЛОВ

Кому адресована эта статья

Несмотря на то что первые автоматизированные информационные системы стали появляться уже более 30 лет назад, среди людей, далеких от программирования, до сих пор бытуют довольно туманные представления о том, что же эти системы из себя представляют. На вопрос о том, что такое база данных, можно услышать самые разнообразные ответы. Чаще всего считают, что это просто некий большой архив. А те, кто увлекается научной фантастикой, могут сказать, что это огромная библиотека, в которой содержатся чуть ли не все накопленные человеком знания и из которой компьютер каким-то таинственным образом может мгновенно извлечь все что угодно. А если говорят о «банке данных», то под этим подразумевают нечто вроде базы данных, только посолиднее. В результате одни не хотят с ними связываться, предпочитая обходиться обычной картотекой, а другие — потому что «это дело темное».

Широкое развитие средств вычислительной техники привело к ее проникновению практически во все сферы человеческой деятельности. Появление надежных, простых в эксплуатации и доступных персональных компьютеров позволяет рациональнее организовать свою работу огромной армии людей, связанных с хранением и обработкой больших массивов данных. Автор надеется, что эта статья поможет людям самых разных специальностей лучше овладеть современной технологией оперативной обработки информации.

Что такое база данных

Многие вещи нам не понятны не потому, что наши понятия слабы; но потому, что сии вещи не входят в круг наших понятий.

Козьма Прутков
«Мысли и афоризмы»

Безусловно, ценность информации люди понимали с незапамятных времен. Достаточно вспомнить Шахерезаду, для которой своевременная и строго дозированная передача информации имела жизненно важное значение. Поэтому естественно возникло желание накопить информацию, сохранить ее и как-то упорядочить, чтобы облегчить доступ к ней и иметь возможность воспользоваться необходимой ее частью.

Говорят, что наука начинается с систематизации. Наверное, можно сказать шире: любое активное освоение окружающего мира начинается с систематизации наблюдаемых объектов, явлений и отношений между ними. Среди традиционных способов упорядоченного хранения нужных сведений можно назвать записную книжку (как говорят, «протез для памяти»), словарь, настольную картотеку, библиотечный каталог. Повсеместное применение компьютеров среди многих других областей не минуло и эту. В последнее время бытует даже расхожее выражение «компьютерная революция», к которой привело использование ЭВМ в области управления данными. Но употребление термина «рево-

люция» здесь по меньшей мере спорно. Ведь внедрение ЭВМ фактически привело к закреплению традиционных методов работы с данными, позволив лишь существенно усовершенствовать технологию и увеличить скорость их обработки. Эти усовершенствования выражаются в более рациональной организации структур данных, увеличении объемов легко доступной информации, сокращении времени поиска и в ряде дополнительных удобств. Вот если бы компьютеры не пришли на помощь человеку, тогда, по-видимому, действительно пришлось бы прибегнуть к каким-то революционным преобразованиям, чтобы не захлебнуться во всевозрастающих потоках информации.

Все это означает, что при разработке любых баз данных (БД) нужно прежде всего исходить не только из интересов конечного пользователя, но и как можно теснее увязывать структуру проектируемой БД с представлением о ней предполагаемого пользователя (или группы пользователей), учитывая то, что он является специалистом в своей предметной области, и, следовательно, структура данных, с которой он привык иметь дело, по-видимому, близка к наиболее рациональной.

Как и в любой другой специальной области, в начале знакомства с БД и СУБД необходимо ввести несколько общих определений. Следует сказать, что терминология в рассматриваемой области до сих пор окончательно не сложилась, и даже в специальной литературе у разных авторов встречаются несколько отличные друг от друга толкования одного и того же термина. Можно сказать, что терминология «живет» своей собственной жизнью, отображая динамику изменений в соответствующих областях знаний. Но для нас вопрос строгости и однозначности терминов и определений не имеет такого принципиального значения, как, например, в математике. Смысл используемых терминов не всегда очевиден, но это не должно смущать читателя. Известный специалист в области проектирования баз данных Дж. Мартин в одной из своих книг высказался в том смысле, что за мудреными словами стоят, в сущности, довольно простые понятия. В теории баз данных предпринята попытка подобрать термины таким образом, чтобы хоть как-то отобразить суть описываемых ими объектов. Хотя лаконично описать емкое понятие удается, естественно, не всегда.

Итак, что же такое база данных? Обычно под базой данных понимают некоторым образом организованную совокупность данных, которые отображают состояние объектов какой-либо предметной области и отношения между этими объектами. Понятно, что в основе такой организации лежит определенная структура, подчас довольно сложная и громоздкая. Но человеку, использующему БД, вовсе не обязательно знать целиком ее структуру и все типы загруженных в нее данных. И уж тем более ему не нужно знать, где, как и в каком виде физически хранятся те данные, которые он получает. Ведь, скажем, бухгалтера, которому нужны сведения о зарплате сотрудника, совершенно не интересует, записаны ли эти сведения на магнитной ленте, магнитном диске или хранятся в основной памяти компьютера. Не должно его, по-видимому, интересовать, и каким видом спорта увлекается данный сотрудник, и есть ли информация об этом в базе данных. Так как мы здесь говорим в основном о компьютерных базах данных, то понятно, что вся работа с данными происходит с помощью программ. Различные программы могут по-разному «видеть» (а следовательно, и использовать) одни и те же данные. Тем самым для любой программы все данные, кроме своих, «прозрачны», т. е. она их просто не замечает.

Для того чтобы несколько пользователей могли получать нужную каждому из них информацию, не мешая друг другу и не затрачивая на эту процедуру слишком много усилий, существуют системы управления базами данных. Они, по сути, представляют собой программы, которые обеспечивают всю работу с базой данных: создание, ведение и совместное исполь-

зование БД многими пользователями. Далее мы рассмотрим каждый из этих аспектов работы СУБД подробнее, а сейчас просто перечислим задачи, которые ей при этом приходится решать.

Для того чтобы создать БД, нужно, естественно, определить, какие данные в ней будут храниться и какого они типа (т. е. будут ли это числа, над которыми предстоит выполнять арифметические действия, текстовая информация или какая-то другая, например графическая). Кроме того, хотя информационная емкость современных компьютеров во много раз выше, чем, например, библиотечных каталогов, она тоже не безгранична. Поэтому желательно с самого начала позаботиться об ограничении объема хранимых данных. Для этого следует задать каждому элементу информации конкретную длину, ограничив ее разумными пределами. Для описания всех этих свойств данных СУБД обычно использует свой собственный язык, так называемый язык описания данных (ЯОД), и после того как данные описаны, принимает решение об их размещении в памяти. Кроме ЯОД, есть и другой способ общения пользователя и СУБД, при котором программа задает вопросы, касающиеся наименования, типа и размера каждого элемента данных, а пользователь отвечает на эти вопросы в режиме диалога с ЭВМ. Общие описания данных иначе называют метаданными.

После того как закончено описание всех элементов будущей БД, начинается формирование ее структуры, т. е. определяются связи между элементами. Подробнее мы еще поговорим об этом, а сейчас отметим, что обычно СУБД может поддерживать структуру лишь какого-то определенного типа, например «древовидную» структуру или «сетевую». Все описания элементов и структуры заносятся в память ЭВМ, и СУБД обращается к ним по мере надобности.

Следующая задача, решаемая СУБД, — обеспечение ввода данных в память компьютера, или, как говорят, загрузка базы данных. На этом этапе СУБД обычно осуществляет контроль правильности вводимой информации. Это может быть, например, проверка данных по типу. Так, если вместо числа вы попытаетесь ввести буквы, программа откажется переслать их в память и выведет на экран дисплея соответствующее сообщение. Возможны и другие, более тонкие методы контроля. При загрузке базы данных большого объема такие процедуры совершенно необходимы, так как они помогают заранее выявить значительную часть неизбежных при вводе ошибок. Размещением вводимой информации в памяти компьютера также управляет СУБД, используя описания элементов и схемы БД. Как уже упоминалось, пользователю об этом заботиться не нужно. Процедура размещения записей в физической памяти «прозрачна» для пользователя. На этом процесс создания БД можно считать законченным.

А какие задачи возникают перед пользователем при эксплуатации БД? Прежде всего это задача обновления базы данных, т. е. замены устаревших данных новыми и добавление свежей информации. Эту задачу актуализации БД также помогает решать СУБД. Затем пользователь (а в больших системах — целая группа пользователей) стремится выбрать из всех хранимых данных только те, которые ему необходимы в данный момент. Свой запрос он обращает в СУБД. Для этого можно использовать либо специальный язык запросов (ЯЗ), который близок к естественному языку, либо другие средства. О них отдельно мы поговорим позже, а сейчас лишь отметим, что все они существенно упрощают процедуру выборки нужной информации.

Если базой данных одновременно могут пользоваться несколько человек, то СУБД должна позаботиться о том, чтобы они не мешали друг другу. Иначе может возникнуть, например, такая ситуация, когда один пользователь желает получить из хранилища какой-то элемент данных, а другой в это же самое время начинает этот элемент менять.

Могут возникнуть и другие ослож-

нения, связанные с одинаковыми запросами. Современные СУБД ко всеобщему удовольствию могут разрешать такие конфликтные ситуации. Кроме того, СУБД должна «уметь» сохранить основную информацию при неожиданном отключении электропитания или машинном сбое. Этот круг задач называется обеспечением целостности базы данных.

Данные, извлеченные из хранилища, обычно подвергаются какой-либо обработке. Один из наиболее употребимых видов такой обработки — сортировка. Например, числа можно сортировать по убыванию или по возрастанию, а строки символов — в алфавитном порядке. Нередко приходится производить объединение элементов или целых блоков. Например, несколько слов можно объединить в предложение. Это, наконец, может быть и чисто математическая обработка, такая, как подсчет суммы или среднего значения нескольких чисел. СУБД должна обеспечить возможность такой обработки.

И наконец, полученную информацию обычно нужно оформить таким образом, чтобы она была представлена в доступной и наглядной форме. Это может быть, например, таблица, снабженная заголовками, поясняющими надписями и разграничивающими рамками. Или это могут быть график, диаграмма или гистограмма с цифровыми данными по осям и подрисуночными подписями. Вывод таких таблиц или графиков должен производиться как на экран дисплея для оперативного просмотра, так и на печать для получения так называемой твердой копии. Такие процедуры выполняет генератор отчетов, входящий в состав СУБД.

Теперь скажем несколько слов о том, что принято понимать под термином «банк данных». Банком данных называют систему программных, языковых, организационных и технических средств, предназначенных для централизованного накопления и коллективного использования данных. В состав банка данных входят базы данных и системы управления базами данных. В банке есть еще архивы, которые вместе с базами образуют фонд данных.

После всего сказанного у читателя может сложиться двоякое впечатление. С одной стороны, автор утверждает, что использовать современные СУБД весьма просто, а с другой — что для этого необходимо научиться проектировать базы данных со сложной структурой, да попутно еще освоить несколько специальных языков и подсистем СУБД. Попробуем разобраться, в чем тут дело. Во-первых, следует различать базы данных на больших ЭВМ, предназначенных для коллективного использования, и базы данных на персональных компьютерах, рассчитанные, как правило, на узкий круг пользователей или на индивидуальную эксплуатацию. Разработка и создание БД первого типа действительно является сложной задачей, которую решает довольно большой коллектив разработчиков, проектировщиков и программистов. Усилиями прикладных программистов (разработчиков прикладных программ) для пользователя создаются максимальные удобства. Для этого заранее определяются, какие данные и в какой форме будут предоставлены потребителю. В случае изменения требований пользователя к информации, получаемой из БД, может потребоваться доработка системы и создание новых прикладных программ.

Разработка же баз данных на микро-ЭВМ обычно преследует гораздо более скромные цели как в плане структуры хранимых сведений, так и в отношении их объема. Поэтому здесь можно использовать более простые (и что немаловажно, гораздо более дешевые) средства, которые могут обеспечить все потребности пользователя и без программирования. Причем возможности таких систем не так уж малы. Имея в своем распоряжении такую систему, можно создать, например, базу данных, содержащую все сведения, хранящиеся в регистратуре районной поликлиники. Эти сведения можно очень просто менять и пополнять. Существенно сократится

время, затрачиваемое на поиск нужных данных, составление и распечатку всевозможных справок и таблиц, статистической отчетности. Иными словами, основную часть утомительной канцелярской работы можно будет переложить на ЭВМ.

Любители логических заключений могут вывести из этого мораль, и даже две. Во-первых, современный пользователь базы данных может прекрасно обойтись без программистов. Во-вторых, предоставить ему эту возможность могут только программисты.

Такие системы уже проникли в нашу жизнь, и мы пользуемся их услугами, подчас не замечая этого. Так, например, приобретая билеты на поезд или на самолет, мы прибегаем к помощи банка данных автоматизированной системы «Сирена». Кассир авиакассы обычно не относится к программистам, тем не менее во время оформления очередного билета он использует большинство основных возможностей СУБД. Система отвечает на запрос о наличии свободных мест, осуществляя поиск в БД, после покупки билета изменяет содержимое БД, актуализируя ее, и наконец, формирует отчет, впечатывая необходимые данные в бланк билета. Системы попроще работают на некоторых техцентрах по обслуживанию автомобилей. Они используют небольшие ЭВМ типа «Искра». Массовый выпуск персональных компьютеров позволяет создавать небольшие базы данных в каждой организации и в отдельных подразделениях предприятий, институтов и учреждений.

Уже появились в продаже бытовые компьютеры, и недалек тот день, когда в наших квартирах будут создаваться «домашние» БД, в которых можно будет хранить и каталог личной библиотеки, и сведения о днях рождения друзей и знакомых, и рецепты приготовления любимых блюд, и многое, многое другое.

Первая часть этой статьи посвящена вопросам разработки, создания и функционирования баз данных. В ней расшифровываются некоторые специальные термины и описаны самые общие принципы проектирования баз данных, различные способы организации их структуры. Читателям, которые будут создавать свои небольшие БД на персональных компьютерах, это поможет ознакомиться с общим подходом к построению базы данных для конкретной предметной области. Далее в статье описываются простейшие способы доступа к хранимой информации и приводятся некоторые сведения по языкам запросов БД. Здесь же рассказано, какие основные манипуляции над данными обычно выполняются с помощью СУБД на персональной ЭВМ и какую пользу можно извлечь из активного взаимодействия с базой данных.

Структура базы данных

Нет столь великой вещи, которую не превзошла бы еще бóльшая. Нет вещи столь малой, в которую не вместилась бы еще меньшая.

Козьма Прутков
«Мысли и афоризмы»

Для того чтобы создать свою собственную базу данных, нужно прежде всего решить, из каких элементов она будет состоять. Здесь можно отчасти положиться на свой повседневный опыт. Например, если вы хотите создать электронную копию личной записной книжки, то, естественно, включите в нее фамилии, имена и отчества своих знакомых, их адреса, телефоны и т. д. Из этих элементов можно составить запись, т. е. объединить их так же, как это обычно делается на бумаге. Объединяя элементы в запись, мы устанавливаем между ними определенную связь. Наличие таких связей и определяет структуру базы данных. Если хранилище информации предназначено для накопления сведений, например о родословной чистокровных животных, то наиболее естественной выглядит древовидная структура, обычно изображаемая в виде генеалогического дерева. Связи такой структуры позволяют для любого предка определить всех его потомков.

Более сложный тип структуры — сетевая, в которой любой элемент БД может быть связан с любым другим. И все-таки наиболее привычным и естественным способом группировки данных является представление их в виде таблицы, состоящей из однотипных записей. Такая структура получила название реляционной. Наверное, к самому замечательному ее свойству относится то, что к ней можно свести любую, самую сложную совокупность данных и связей между ними. Популярность реляционных баз данных с каждым годом неуклонно растет, и на персональных компьютерах используются практически только системы такого типа.

Взгляните на рис. 1, где изображен пример двумерной таблицы, используемой в реляционных БД. Каждая строка образует отдельную запись и состоит из полей определенной длины, имеющих уникальные имена. Вся совокупность однотипных записей образует файл, которому также присваивается свое имя. Обратите внимание на то, что в первом поле (табельный номер) информация от записи к записи не повторяется. Такое поле называется ключевым. Его наличие весьма желательно, так как позволяет однозначно идентифицировать каждую запись файла.

Теперь предлагаем читателю провести небольшой эксперимент, который поможет нагляднее представить некоторые проблемы, возникающие при создании даже самой элементарной базы данных. Попробуйте создать базу данных, состоящую из однотипных записей, которая могла бы заменить вам записную книжку с адресами и телефонами ваших знакомых. Для этого просто выпишите в столбец наименования элементов, составляющих запись. Затем решите, какую длину (в буквах, цифрах или иных символах) будет иметь каждый элемент, и проставьте соответствующую величину против названия элемента. Теперь заполните соответствующие элементы данными из своей записной книжки, выбрав из нее хотя бы три первые записи. А затем ответьте на следующие вопросы.

Табельный	Имя	Дата рождения	Отдел	Должность
121001	Волков А. А.	12.10.47	121	Начальник отдела
121002	Зайцев А. Б.	10.01.50	121	Зав. сектором
121003	Лисицын Б. В.	03.07.42	121	Ст. науч. сотрудник
121004	Козлов О. Т.	07.05.58	121	Мл. науч. сотрудник

Рис. 1

1. Все ли необходимые элементы записи вы учли? Так, для почтового адреса, определили ли вы поля: индекс, республика, область, район? Ведь в некоторые адреса эти элементы могут входить.

2. Предусмотрели ли вы поля под те сведения, которые могут понадобиться вам в будущем? Например, вы обычно не записываете место работы своих знакомых и друзей, но иногда это будет необходимо.

3. Хватило ли отведенного под каждый элемент места для записи его значения целиком?

4. Предусмотрели ли вы возможность изменения длины элементов, например изменения номеров телефонов с семизначных на восьмизначные?

5. Не заняли ли вы место под абсолютно ненужные знаки, например под дефисы в номерах телефонов? Ведь вместо девяти знаков 190—18—92 целесообразно хранить только семь: 1901892.

Таких вопросов можно задать еще множество. И это в самом простом случае! Допустим, вы отвели под фамилию 20 позиций. А есть ли в вашей записной книжке хоть одна фамилия такой длины? И сколько незанятых позиций осталось у вас только в первых трех записях?

А теперь попробуйте что-нибудь предпринять для экономии места в памяти компьютера. Прежде всего следует по возможности уменьшить избыточность данных. Для этого можно определить элементы, у которых наиболее часто повторяются значения, и выделить их в отдельную запись. Большинство ваших знакомых наверня-

ка живут в одном городе. Так зачем же повторять его название в каждой записи? Можно, например, выписать все встречающиеся в вашей записной книжке города, перенумеровать их и создать отдельный файл с записями городов, в котором номер будет играть роль ключа. Тогда в основной записи вместо названия города можно будет хранить только его номер. Теперь обратите внимание на связь между наименованием города и почтовым индексом. Как можно использовать эту связь для сокращения избыточности?

Даже из такого элементарного примера видно, что затраты времени на планирование и организацию БД оборачиваются значительной экономией машинной памяти и в конце концов существенным увеличением эффективности работы с базой данных. Иначе говоря, чем тщательнее продумана структура БД, тем качественнее будет готовый продукт.

Простые в освоении языки базы данных для конечных пользователей

Жизнь и смерть во власти языка и любящие его вкусят от плодов его.

Екклезиаст

Большинство языков запросов к базе данных разрабатывается для конечных пользователей, не имеющих специальной подготовки в программировании. Тем не менее многие из этих языков требуют освоения некоторого синтаксиса и мнемоники и вряд ли могут пригодиться тем пользователям, которые лишь эпизодически обращаются к БД. Но существуют средства, созданные и специально для них.

Мы здесь рассматриваем только диалоговый режим общения пользователя с компьютером, когда пользователь работает с терминалом и получает ответы на свои запросы с минимальной задержкой (от долей до нескольких десятков секунд). В принципе возможен и другой режим работы, когда пользователь представляет свои запросы в группу обслуживания БД на специальных бланках, заполненных в соответствии с установленной формой, и получает ответ через несколько часов. В последнее время за рубежом даже появилось большое количество фирм, осуществляющих посреднические функции по получению потребителями информации из крупных банков данных. Но мы здесь рассмотрим лишь способы быстрого извлечения данных.

Все системы управления базами данных можно условно разделить на три большие группы. Первая использует для манипулирования данными обычные алгоритмические языки, такие, как ПАСКАЛЬ, ПЛ/1 или ФОРТРАН. Вторая — обходится для этих целей своими собственными, «встроенными» в СУБД средствами. И третья — объединяет обе возможности. Большинство СУБД для персональных компьютеров относится ко второму типу. Именно их описанию мы и уделим основное внимание.

Обычно мощные СУБД предоставляют (или должны предоставлять) пользователю следующие возможности:

— доступ к данным с удаленного терминала;

— чтение одним пользователем данных, в то время как другие читают, видоизменяют, вставляют или удаляют записи;

— формирование отчетов из полученных данных;

— специальные виды обработки, такие, как сортировка, индексирование и т. п.;

— создание собственных файлов пользователя;

— обеспечение секретности или ограничения доступа к части информации.

Задачи, решаемые СУБД персональных компьютеров, проще, так как им, например, нет необходимости поддерживать возможность одновременной работы нескольких пользователей с базой данных. Языки, применяемые в таких системах, также можно условно разделить на типы. Одним из наиболее простых в обращении языков запросов является такой, в котором спрашивающий заполняет некоторую форму. Форма высвечивается на экра-

не дисплея и заполняется с клавиатуры, как на обычной пишущей машинке. Реклама одного из таких языков утверждает, что его можно освоить за пять минут!

Автору однажды пришлось быть свидетелем того, как человек осваивал такую систему «на ощупь», не имея под рукой никаких инструкций, используя метод проб и ошибок. И надо сказать, результат был довольно впечатляющим! Для получения отчета по содержимому какого-нибудь файла пользователь набирает на клавиатуре компьютера имя этого файла, после чего система выводит на экран дисплея информацию об элементах записей, составляющих этот файл. Кроме того, на экран выводится форма, которую предлагается заполнить пользователю. Форма эта состоит из нескольких областей. Прежде всего это область, определяющая выбор нужных записей. Для выбора записи можно использовать несколько условий. Простой пример выбора записи по условию показан на рис. 2. Показанное здесь условие можно описать такими словами: «Отобрать все записи, у которых номер =25 и год 〉 1980». Как строится это условие, понятно из рисунка. Добавим лишь, что в графу «Имя элемента или константа» можно заносить имя другого элемента. Тогда будут сравниваться значения этих элементов. Буква Д в графе «Тип» определяет тип данных указанного элемента. В нашем примере это десятичное число.

Выбор можно производить не только по числовым, но и по символьным данным. Под символами подразумеваются буквы и некоторые другие специальные обозначения, такие, как точка, тире и т. п. Таким образом, можно задавать условия, например вида: «фамилия-Петров». При этом система будет искать запись, содержащую в поле «Фамилия» символы «Петров».

Логическая	Имя элемента	Операция	Тип	Имя элемента или константа
	номер	=	Д	25
и	год	>	Д	1980

Рис. 2.

Во второй раздел формы вводится информация об оформлении отчета, который вы хотите получить. В соответствующие графы вводятся имена элементов записи и ширина каждого столбца отчета (каждый элемент печатается в отдельном столбце). Кроме того, можно снабдить отчет общим заголовком, потребовать просуммировать все значения какого-либо столбца, задать поиск минимального или максимального значения и т. п. На этом заполнение фкрмы заканчивается, и после запуска программы отчет выводится на печать.

Нетрудно убедиться, что это на самом деле простой язык и его можно освоить за пять минут, а то и меньше. Но несмотря на свою простоту, он вполне может удовлетворить некоторую категорию пользователей.

Следующая разновидность языков конечного пользователя — «запрос по примеру». Язык запросов должен быть таким, чтобы можно было начать работать с компьютером практически безо всякой подготовки. При заполнении вышеописанной формы нужно лишь понимать сравнительно небольшое число ее заголовков, но разработчики языков стремятся и это число свести к минимуму. Как показывает опыт, человеку, который никогда в жизни не прикасался к компьютеру, достаточно 10—15 минут для того, чтобы научиться использовать язык запросов по примеру, если предварительно ему «в двух словах» объяснить, как организованы плоские файлы и какого рода информацию они содержат.

Как показали специальные исследования, совершенно неподготовленному пользователю требуется меньше трех часов на освоение искусства формирования сложных запросов. В начале работы пользователю на экране терминала предлагается форма таблицы, показанная на рис. 3. В соответствующие места таблицы можно вводить с клавиатуры имя файла, имена элементов (или, как их называют, имена полей), или значения элементов, для того чтобы сформировать запрос. Кроме того, можно изменять данные,

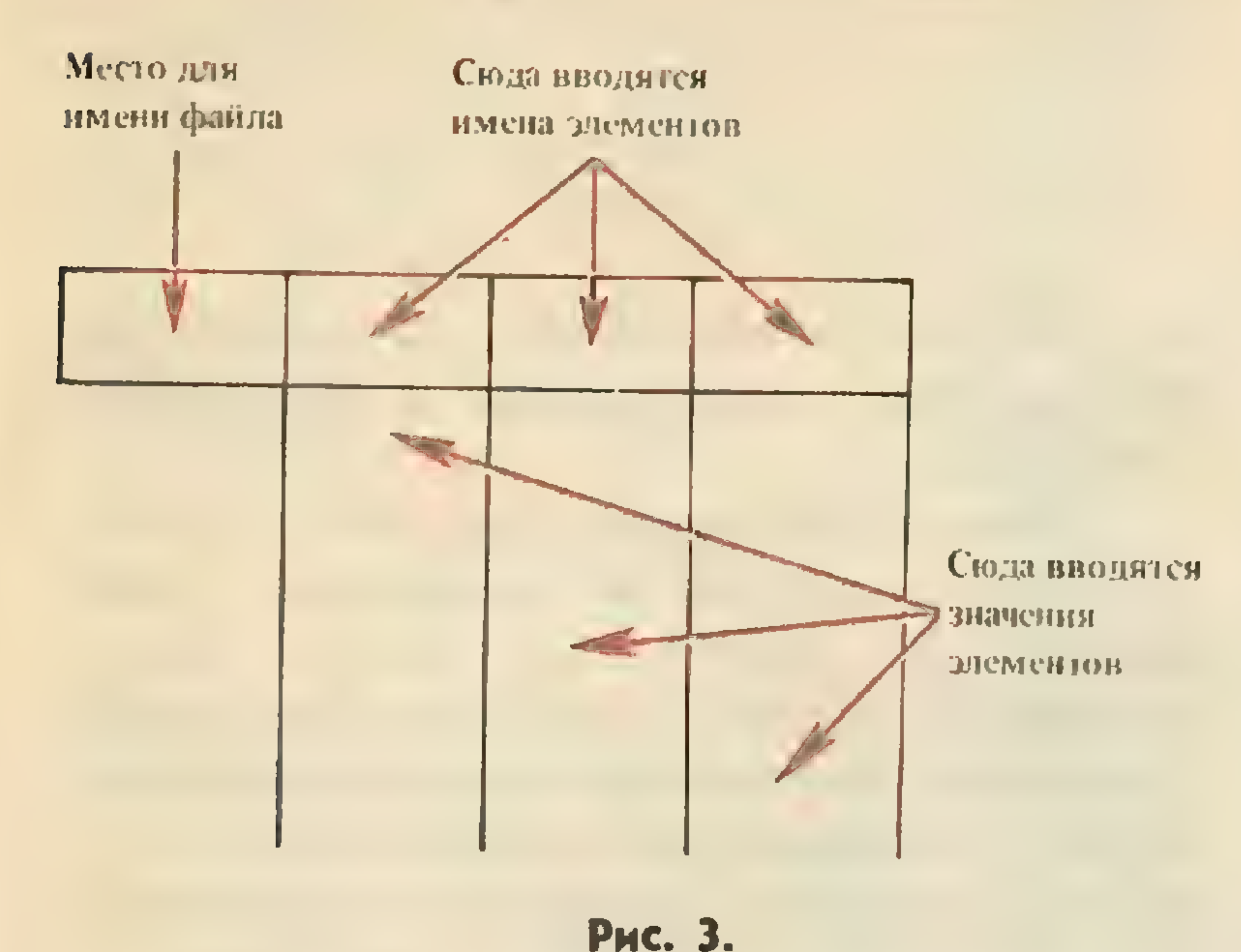


Рис. 3.

уничтожать какие-либо их части или добавлять новые. А если пользователь хочет, чтобы компьютер сам заполнил соответствующее место таблицы, можно просто напечатать на этом месте букву Р. (печать). После буквы — команды ставится точка, чтобы не спутать ее с обычным текстом. Если букву Р напечатать сразу после имени файла, то на экран дисплея автоматически будут введены имена всех содержащихся в нем полей. Допустим, в нашей базе данных хранится файл «кадры», каждая запись которого состоит из четырех полей: номер, имя, отдел и должность. Напечатав в первом окне таблицы «кадры Р.», мы получим на экране такую картинку:

Кадры	Номер	Имя	Отдел	Должность

Если нас интересует список всех сотрудников, работающих в каком-то конкретном отделе, то в графе «отдел» нужно напечатать его номер, а в графе «имя» — букву Р. В графе имя будет напечатан список фамилий всех сотрудников указанного отдела.

В принципе пользователю не обязательно помнить даже имя нужного ему файла. Например, напечатав (в любом столбце) исходной формы слово «отдел», а в первом окне букву Р., мы получим список всех файлов, содержащих поле «Отдел».

Р.	Отдел		

Если просто напечатать в первой графе букву Р, оставив остальные пустыми, то на экран будут выведены имена всех файлов, хранящихся в базе данных.

Таким образом, можно приступать к диалогу с системой, не зная ни языка программирования, ни содержимого базы данных. Можно не знать даже имен файлов и их полей. Вот это и называется «дружественным пользователю» программным обеспечением. Через несколько минут работы с такой системой можно формировать довольно сложные запросы. Например, чтобы построить такой запрос: «Перечислить фамилии всех сотрудников отдела 121, оклад которых превышает 200 рублей и которые имеют больше двух детей», нужно заполнить форму таким образом:

Кадры	Номер	Имя	Отдел	Оклад	Дети
		Р.	121	>200	>2

Для описания условий выбора нужных записей можно использовать обычные символы сравнения: >, <, >= (больше или равно), <= (меньше или равно) и <> (не равно).

После того как на экране появится выведенная из базы данных информация, можно внести изменения в данные. Для этого в первой графе на строке, подлежащей изменению, следует напечатать букву U. (с точкой). При этом можно использовать арифметические действия:

Кадры	Номер	Отдел	Оклад
	121001	121	250
	121004	121	230
U.	121015	121	210*1.1

В результате ввода такой записи число в третьей строке графы «Оклад» будет изменено на 231.

При некоторых более специфических типах запросов пользователь задает пример результата, который он хочет получить. Для этого вместо действительного значения вводится некоторая условная величина. Чтобы показать, что эта величина записана в качестве примера, ее выделяют подчерки-

ванием. По примеру можно изменять значения сразу нескольких величин в указанном поле. Так, после ввода такой строки

Кадры	Номер	Имя	Отдел	Оклад
U.			121	100*1.1

все значения в графе «Оклад» для сотрудников отдела 121 будут увеличены в 1,1 раза. Число 100 введено здесь для примера. Вместо него компьютер будет подставлять реальную величину, хранящуюся в данном поле.

Для выделения примера можно использовать и частичное подчеркивание. Так, например, если в графе «Имя» записать ИВАНОВ, то это будет означать, что буква И — действительная, а остальные могут быть любыми другими. В результате система выведет фамилии сотрудников, начинающихся с буквы И. Аналогично запись 10000 обозначает все числа, заканчивающиеся тремя нулями: 000.

Система допускает и формирование более сложных запросов, когда заполняется несколько строк исходной формы. Эти строки связываются между собой также с помощью примеров. Так, для того чтобы послать системе запрос: «Найти имена всех сотрудников с окладом выше чем у Иванова», нужно заполнить форму следующим образом:

Кадры	Имя	Оклад
	Иванов	200
	Р.	>200

Здесь число 200 — условная величина, вместо которой система подставит действительное значение оклада, соответствующее фамилии «Иванов». Затем будет выполнен поиск всех записей, содержащих в графе «оклад» число, большее найденного. Для проверки полученного результата можно послать другой запрос. Допустим, в отчет на предыдущую посылку была получена фамилия «Петров». Тогда, введя в форму такие строки:

Кадры	Имя	Оклад
	Петров	Р.
	Р.	

вы получите распечатку, содержащую оклады всех сотрудников с фамилией «Петров».

После непродолжительной практики можно еще более усложнить условие поиска. Например, чтобы сформировать запрос: «Есть ли сотрудник с окладом большим, чем у Иванова и Петрова, вместе взятых?», нужно заполнить форму следующим образом:

Кадры	Имя	Оклад
	Петров	100
	Иванов	200
	Р.	>100+200

Естественно, что вместо условных величин 100 и 200 можно использовать и другие обозначения, например А и В. Объединение нескольких строк через условную величину позволяет строить выражения с логическими связками И или ИЛИ. Например, запрос:

Кадры	Имя	Оклад
	Р. ААА	>100
	ААА	<200

будет интерпретирован так: «Вывести фамилии всех сотрудников, оклад которых больше 100 и меньше 200 рублей».

А как поступить, если нужная информация располагается в записях разных типов (т. е. попросту в разных файлах)? Запрос по примеру позволяет обработать и такую ситуацию. Иными словами, записи из разных файлов можно объединять. Пользователь имеет возможность вывести на экран дисплея две (или больше) исходные формы. Используя условные значения (примеры), можно объединить записи, подсоединенные к этим формам. Если кроме файла «кадры» в базе данных есть файл «отделы» с полями «номер» и «начальник», то можно запол-

нить две формы следующим образом:

Отделы	Номер	Начальник
	121	AAA

Кадры	Имя	Оклад
	AAA	P.

При этом в файле «Отделы» будет найдена фамилия начальника отдела 121 и из файла «Кадры» выведен его оклад. Принцип обработки таких запросов, заложенный в языке «запрос по примеру», очень просто понять, так как он отражает обычную последовательность действий при поиске нужной информации в таблицах без помощи компьютера.

Для выполнения некоторых наиболее часто встречающихся операций обработки данных в языке «запрос по примеру» предусмотрен ряд стандартных «встроенных» функций. Это функции

SUM — вычисления суммы;
CNT — подсчета числа записей;
AVG — расчета среднего значения;
MAX — поиска максимального значения;
MIN — поиска минимального значения;
UN — отбрасывания повторяющихся значений.

Для вычисления среднего оклада сотрудников отдела 121 нужно заполнить исходную форму таким образом:

Кадры	Отдел	Оклад
	121	P. AVG

Узнать фамилию сотрудника, имеющего максимальный оклад, можно так:

Кадры	Имя	Оклад
	P.	MAX

Кроме исходных форм, пользователь может вывести на экран еще одну — форму задания условий. Ее можно вызвать в любой момент диалога с компьютером. Используется она тогда, когда условие сложно записывается в обычной исходной форме. Например, запрос: «Вывести фамилии всех сотрудников, работающих в отделе 121 и 122 и имеющих оклад не ниже 150» можно задать следующим образом:

Кадры	Имя	Отдел	Оклад
	P.	ОТД	ОКЛ

Условия
ОТД = (121/122) ОКЛ = >150

В этой форме можно таким образом перечислить целый ряд условий, относящихся к данному запросу.

В процессе работы с базой данных рано или поздно возникает необходимость удаления некоторых записей и ввода новых данных. Как выполняются эти действия? По тому же образцу, что и перечисленные выше. Для обозначения ввода используется символ I., а для удаления — D. (после буквы нужно не забывать ставить точку). Чтобы добавить новую запись к файлу «кадры», нужно вызвать его исходную форму и заполнить соответствующие графы:

Кадры	Имя	Отдел	Оклад
I.	Зуев Д.	121	150

Чтобы удалить эту запись, нужно ввести:

Кадры	Имя	Отдел	Оклад
D.	Зуев Д.	121	150

При этом следует соблюдать осторожность, так как если, например, ввести:

Кадры	Имя	Отдел	Оклад
D.		121	

то будут удалены все записи, содержащие в графе «Отдел» число 121.

Язык «запрос по примеру» предоставляет пользователю еще одну возможность обработки данных, к которой очень часто приходится прибегать на практике. Речь идет о сортировке. Ее можно выполнять по одному или по нескольким полям. Причем данные, содержащиеся в этих полях, не обязательно должны быть числовыми. Числовые данные при сортировке располагаются по убыванию или возрастанию значений чисел, а текстовые данные — в алфавитном или обратном порядке. Для сортировки в возрастающем порядке используются символы AO (ASCENDING ORDER), для обозначения убывающей последовательности — DO (DESCENDING ORDER). В скобках после этих символов можно указать, какая сортировка является первичной, а какая — вторичной. Так, например, если мы хотим расположить записи файла «Кадры» в порядке убывания номеров отделов, а внутри каждого отдела список фамилий сотрудников упорядочить по алфавиту, то исходную форму следует заполнить таким образом:

Кадры	Имя	Отдел	Оклад
P.	AO(2).	DO(1).	

При этом записи на экран будут выводиться с учетом заданных последовательностей сортировки.

В качестве упражнения попробуйте решить такую задачу. Как, используя условные величины (примеры), создать новый файл и занести в него записи из файла «Кадры», отсортированные по возрастанию номеров отделов, а внутри каждого отдела по уменьшению оклада?

В заключение отметим, что существует много разновидностей легких в изучении языков конечных пользователей. Они могут отличаться способами оформления запросов, большими или меньшими возможностями в обработке данных и т. п. Но, во-первых, любой из них действительно можно освоить за несколько десятков минут и, во-вторых, они позволяют выполнять большинство основных операций, необходимых для работы с базой данных. Как показывает опыт, существует огромное количество пользователей, все потребности которых по запросам к базе данных могут быть удовлетворены с помощью этих языков.

dBASE III — система управления базами данных для персональных ЭВМ

Выбирайте выражения!
Из разговора

В последнее время разработано уже несколько десятков СУБД для персональных компьютеров. И каждый год появляются новые системы этого типа. Каждая новая система (или новая версия уже известной) предоставляет пользователю какие-то дополнительные возможности или удобства в работе. Во многом их бурное развитие обусловлено совершенствованием самих персональных компьютеров. Экран компьютера стал цветным — и это дало возможность нагляднее представлять информацию. Увеличился объем оперативной памяти — стало возможным увеличить сложность структуры баз данных. Увеличилось быстродействие — возросли возможности формирования различных запросов и манипулирования данными. Появились сети персональных ЭВМ — возникла необходимость в обмене данными между ними и т. д. Но в основном большинство современных СУБД для персональных компьютеров имеют очень много общего. Так, для кодирования символов в них используется, как правило, один и тот же код. Он называется ASCII (американский стандартный код для обмена информацией). Данные хранятся в плоских файлах. Каждый файл состоит из однотипных записей фиксированной длины и т. д.

Естественно, что много общего имеют между собой и способы работы с разными СУБД. Этого требует как принцип «максимальной дружественности пользователю» — один из основополагающих при разработке, так

и соображения преемственности программного обеспечения — далеко не всякий пользователь захочет приобрести такую систему, которую ему придется заново изучать от начала до конца.

Мы рассмотрим основные способы работы с одной из наиболее популярных систем такого класса — СУБД dBASE III. Первая причина такого выбора заключается в поистине удивительной популярности этой системы. С этой популярностью вынуждены считаться и разработчики нового программного обеспечения. Для того чтобы новая программа пользовалась спросом, часто требуется, чтобы она могла обрабатывать файлы dBASE. Вторая причина нашего выбора состоит в достаточной типичности пакета программ dBASE. Разобравшись в том, как работает эта система, можно получить представление о большинстве основных способов обработки данных, реализуемых СУБД, что в дальнейшем позволит легче освоить какую-либо другую программу. Кроме того, СУБД dBASE III совместима с предыдущей версией — dBASE II. Существует ее новая, расширенная версия — dBASE III+. Имеется ряд версий этого пакета, разработанных у нас в стране. Основные принципы и формы работы с этими системами одинаковы. Здесь удалось почти полностью соблюсти преемственность последующих версий от предыдущих. Каждая последующая версия может использовать файлы данных, созданные предыдущими. И лишь для использования в новых версиях старых программ могут потребоваться незначительные переделки.

dBASE III, вообще говоря, представляет собой пакет взаимодействующих программ, но в дальнейшем для краткости мы иногда будем просто говорить «программа dBASE III». Эту программу можно устанавливать на персональных компьютерах, работающих под управлением операционной системы MS — DOS. Например, на отечественные ЭВМ ЕС-1840, ЕС-1841, болгарские компьютеры «Правец», компьютеры фирмы IBM или совместимые с ними. Способы работы с dBASE от типа компьютера не зависят.

Основным объектом обработки этой СУБД являются файлы данных. Каждый такой файл состоит из однотипных записей. Число записей в файле практически не ограничено (теоретически его можно довести до миллиарда, хотя и трудно предположить, что кто-либо в течение своей жизни сможет создать такую базу данных). Но максимальная длина каждой записи ограничена — она не может превышать 4000 символов. Впрочем, для большинства приложений такая длина вполне достаточна. Запись состоит из полей (их можно рассматривать как столбцы плоской таблицы). Каждому полю присваивается свое имя. На имя поля, так же как и на имя файла, наложены ограничения. Имя файла не должно превышать в длину восьми символов, а имя поля — десяти, причем каждое имя должно начинаться с буквы. В каждом файле данных разрешается определять до 128 полей, а каждое такое поле может содержать до 254 символов, но при этом, естественно, общее число символов во всех полях не должно быть больше 4000 — максимальной разрешенной длины всей записи.

Вся эта организация в принципе ничем не отличается от той, что мы рассматривали выше. Но в dBASE значительно расширены возможности манипулирования данными. Одно из средств такого расширения — разделение данных на различные типы. При определении полей файла данных каждое поле может быть отнесено к одному из следующих типов:

1. Поле календарной даты. Обозначается символом D(DATE).

2. Символьный тип. Обозначение — C (CHARACTER).

3. Числовой тип. Обозначение — N (NUMBER).

4. Логический тип. Обозначается символом L(LOGICAL).

5. Поле примечаний. Обозначение — M (MEMO).

Каждый тип поля предназначен для хранения специфической информации.

В поле D, как это следует из названия, можно хранить только цифры, обозначающие дату. В поле С можно записывать любые имеющиеся на клавиатуре символы: буквы, цифры, точки, запятые и т. д. В числовое поле можно заносить только цифры, знаки + и — и десятичную точку. Поле логического типа содержит величины, которые могут принимать только одно из двух значений: «истинно» или «ложно». В поле примечаний можно ввести любой текст.

Но для чего нужно все это разнообразие? Прежде всего для подстраховки от ошибок. Так, например, если вы по забывчивости или невнимательности захотите ввести в числовое поле какие-то буквы, система откажется их воспринять. То же самое произойдет, если вы попытаетесь данные из символьного поля использовать в математических выражениях. Особую и очень полезную роль играет поле типа MEMO. Допустим, вы создали файл данных, включающий в себя 1000 записей (например, для картотеки поликлиники). В какой-то момент вам понадобилось записать дополнительные сведения о десяти больных. Как при этом поступить? Добавить к каждой записи новое поле? Но тогда в 990 записях это поле останется пустым, а это означает, что память ЭВМ используется очень неэффективно. Поле MEMO существует для разрешения именно таких ситуаций. Записи поля MEMO хранятся отдельно от файла данных и занимают место в памяти только тогда, когда в них внесена реальная информация. Ряд дополнительных удобств дает и использование полей типа D и L.

Описания всех имен, типов и размеров полей составляют структуру файла, которая хранится в памяти вместе с данными. Эту структуру всегда можно просмотреть и при необходимости исправить.

Кроме файлов данных, dBASE использует ряд других типов файлов. Это прежде всего файлы, содержащие наборы команд — «командные файлы». Существуют и специальные файлы, в которых хранятся формы вывода информации на экран дисплея или на принтер — форматные файлы, и ряд других, о которых будет рассказано ниже. Для того чтобы различать типы файлов, dBASEIII, так же как и операционная система, добавляет к их имени точку и три символа, которые называются «расширением» имени. Файлы данных в dBASEIII имеют расширение .DBF, командные файлы — .PRG и т. д. Так что если вы встречаете файл с именем КАДРЫ.DBF, то это, по всей видимости, файл данных, созданный dBASE.

Как уже упоминалось, dBASE III предоставляет пользователю богатые возможности в отношении манипулирования данными. Реализуются они с помощью команд собственного языка системы. Но при этом в системе сохранена возможность работы без непосредственного использования языка команд. Это работа с помощью «меню», в которых предлагаются различные возможности, а пользователь выбирает какую-либо из них. Такой режим работы освоить не сложнее, чем язык запросов по примеру, и пользователи, не имеющие времени на изучение языка команд или не нуждающиеся в какой-либо сложной обработке информации, вполне могут ограничиться знакомством с этой частью системы.

Режим работы с активным использованием меню называется режимом ASSIST (режим содействия). Он устанавливается автоматически при первоначальном вызове программы dBASE. Если пакет dBASE III хранится на твердом магнитном диске вашего компьютера, то для начала работы с ним достаточно набрать на клавиатуре символы DBASE и нажать клавишу «ввод». На экране дисплея появится первое основное меню. В этом меню перечислены возможности, предоставляемые пользователю системой. Выбор осуществляется с помощью клавиш со стрелками. Возможность, предоставляемая в текущий момент, выделена на экране «негативным» изображением, т. е. она обозначается темными буквами на ярком фоне. Если нажать клавишу «стрелка вправо», то яркое

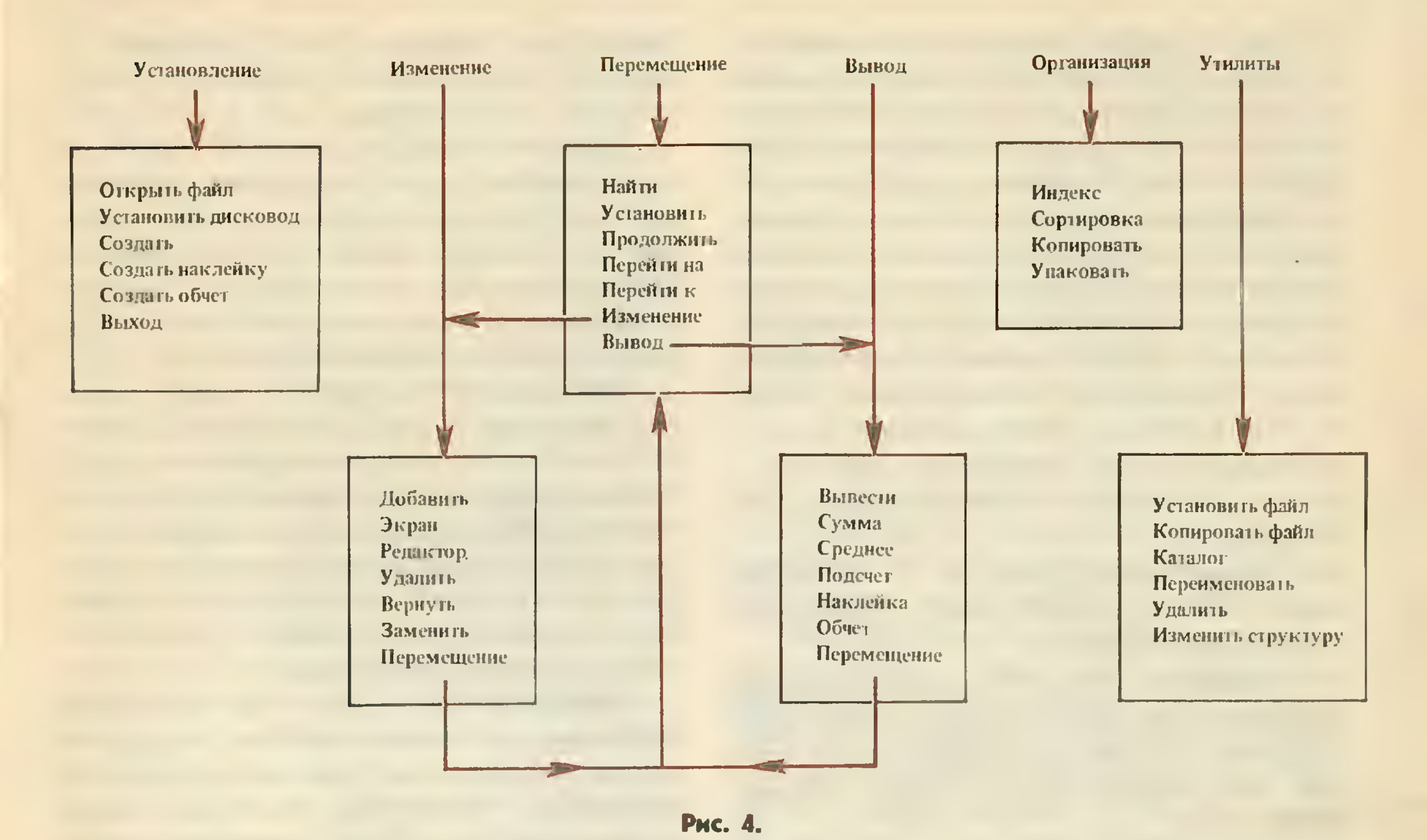


Рис. 4.

окошечко переместится на следующую надпись справа, нажимая клавишу «стрелка вниз», можно переместить окошечко на одну строчку вниз и т. д. После того как выбрано желаемое действие, нужно нажать клавишу «ввод».

Общая схема использования основного меню показана на рис. 4. В начале работы на экране дисплея появляется первая строка меню, ярко высвечивается первое слово «установ» и под ним записаны варианты использования этой возможности. Если нажать клавишу «стрелка вправо», то высветится слово «изменение» и под ним в рамке соответствующие функции этого режима работы. Если нажать клавишу «стрелка вниз», то светящее окошко переместится вниз, позволяя осуществить выбор нужной функции.

Рассмотрим вкратце некоторые из предлагаемых возможностей работы.

Открыть файл. Прежде чем начать работу с файлом данных, необходимо указать системе, какой именно файл вы хотите использовать. Если на экране ярко светится окошко со словами «открыть файл» и вы нажмете клавишу «ввод», то система предоставит вам список имен всех файлов данных, хранящихся в памяти. Действуя так же, как и раньше, т. е. перемещая яркое окошко по экрану с помощью клавиш-стрелок, можно выбрать нужный файл и, нажав клавишу «ввод», «открыть» его, т. е. получить возможность доступа к записанным в нем данным.

Установить дисковод. Почти все современные персональные компьютеры имеют несколько дисководов. Нажав клавишу «ввод», вы можете указать системе, в каком дисководе находится диск с вашими файлами, и в дальнейшем программа будет работать с этим диском.

Создать. Эта команда служит для создания нового файла данных. О процедуре работы с ней мы расскажем немного позже.

Создать наклейку. Существует специальная форма вывода данных на печать, часто помогающая избежать рутинной работы. Если, например, вы хотите заготовить ряд наклеек на почтовые конверты так, чтобы текст на всех наклейках был одним и тем же, а адрес каждого корреспондента менялся, то с помощью рассматриваемой команды можно сформировать текст такой наклейки и указать про-

грамме, в каком файле данных хранятся адреса. Тогда при распечатке адреса корреспондентов будут выводиться в соответствующее место формы, а остальной текст будет неизменным.

Создать отчет. Эта команда позволяет выбрать из файла данных нужные записи и поля и представить их на экране или на бумаге в виде столбцов, имеющих соответствующие заголовки, т. е. сформировать простейший отчет по содержимому базы данных.

Выход. Для окончания работы с системой можно использовать эту команду.

В следующем режиме — «изменение» можно использовать нижеперечисленные функции.

Добавить. По этой команде можно добавлять новые записи к открытому в данный момент файлу данных. О том, как она используется, будет рассказано ниже.

Редактор. Эта команда служит для редактирования записей. Если нажать клавишу «ввод», когда в ярком окне находится слово «редактор», то на экран будет выведена одна запись открытого в текущий момент файла данных. Содержимое каждого поля записи будет выводиться с новой строки. В режиме «редактор» можно изменять информацию, содержащуюся в отдельных полях, или вводить новую в пустые поля. Затем можно перейти к другой записи.

Экран. Эта функция позволяет редактировать записи в режиме «полного экрана». Ее отличие от предыдущей заключается в том, что записи на экран выводятся подряд, без разделения полей. Изменять нужное место записи можно простым вводом нужных символов с клавиатуры. Иногда такой режим работы удобнее, чем предыдущий.

Удалить. Команда служит для удаления ненужных записей. Задействовав эту функцию, можно указать системе, какие записи подлежат удалению. Но программа dBASE по этой команде только помечает соответствующие записи, а не стирает их сразу из физической памяти. Для того чтобы на самом деле уничтожить запись, используется описанная ниже команда «упаковать».

Вернуть. Если по какой-то причине вы передумали и не хотите уничтожать записи, помеченные для удаления, то команда «вернуть» восстановит первоначальный вид файла — немаловажная возможность для любителей принимать поспешные решения!

Заменить. При работе с числовыми данными может возникнуть необходимость изменить определенным образом содержимое одного поля у всех записей. Ну, например, увеличить оклад всех сотрудников в полтора раза. Данная команда позволяет не перебирать записи по одной, а автоматически изменить их все.

Перемещение. Для организации перехода от одной записи к другой внутри файла данных используется концепция **указателя.** Система запоминает номер записи, с которой работает пользователь, или, как говорят, устанавливает указатель на эту запись. Поэтому если вы отредактируете какую-либо запись, затем выйдете из режима редактирования и снова возвратитесь к нему, то на экране окажется та же самая запись, с которой вы работали. Для поиска нужной записи и перехода от одной записи к другой используются функции режима «перемещение».

Команды **«найти»** и **«установить»** помогают отыскать нужную запись в файле данных (установить указатель на эту запись) по какому-то признаку. Например, можно поставить задачу: найти в файле первую запись, содержащую в поле «имя» символ В. Программа будет последовательно проверять записи файла, пока не дойдет до той записи, у которой текст в поле «имя» начинается с буквы В. Действие команд «найти» и «установить» несколько различно. Дело в том, что команда «найти» обрабатывает только так называемые индексированные файлы. О том, что они из себя представляют, сказано ниже.

Продолжить. По этой команде продолжается поиск следующей записи, удовлетворяющей заданному условию.

Перейти на. Позволяет переводить указатель на одну или несколько записей вперед или назад.

Перейти к. В отличие от предыдущей команды эта устанавливает указатель на конкретную запись. Если вы выбрали эту команду и нажали клавишу «ввод», на экране дисплея появится надпись: «Введите номер записи:». Набрав на клавиатуре соответствующие цифры и еще раз нажав «ввод», вы установите указатель на нужную запись.

Две последние строки **«изменение»** и **«вывод»** служат для перехода в соответствующий режим, как это показано на схеме.

Режим **«вывод»** предоставляет возможность вывода информации на экран. Для того чтобы вся выводимая на экран информация передавалась также на принтер, нужно использовать клавиши ⟨CTRL⟩ и ⟨P⟩. Если нажать клавишу ⟨CTRL⟩ и, удерживая ее в этом положении, нажать клавишу ⟨P⟩ (предварительно включив принтер), то вся появляющаяся на экране информация будет выводиться на печать. Для отмены вывода на печать нужно еще раз нажать ⟨CTRL⟩ и ⟨P⟩.

Вывести. Результатом действия этой команды будет вывод на экран (или на принтер) текущей записи, т. е. той, на которую в данный момент установлен указатель.

Сумма. В ответ на посылку этой команды система предложит вам ввести имя числового поля, в котором требуется просуммировать числа. Результат действия команды — сумма чисел в столбце таблицы.

Среднее. По этой команде вычисляется среднее значение в столбце какого-либо поля.

Подсчет. Выбор этой команды приводит к подсчету числа записей, удовлетворяющих заданному условию. В отчет на посылку команды система предложит вам ввести условие для отбора записей, а затем выведет на экран число записей, удовлетворяющих указанному условию.

Наклейка. Команда инициирует формирование вышеупомянутых почтовых наклеек.

Отчет. По этой команде заготовленный заранее отчет выводится на печать или на экран дисплея. Здесь следует иметь в виду, что в памяти компьютера хранится лишь форма отчета, т. е. имена полей, ширина столбцов, тексты заголовков и т. п. Сами данные для отчета выбираются из файла данных. Поэтому если изменить содержимое файла данных, то изменится и содержание отчета при неизменной форме.

Последняя строка служит для перехода в режим «перемещение».

Следующий по порядку режим, как явствует из его названия, служит целям организации записей в файле.

Сортировка. Выполнение операции сортировки — одна из наиболее употребимых форм организации данных. Если вы выберете эту возможность, система попросит вас ввести имя поля, по содержимому которого будет происходить упорядочение записей. Кроме того, можно выбрать прямой или обратный порядок сортировки. Если поле, по которому выполняется сортировка, символьного типа, то записи располагаются в алфавитном порядке. При совпадении первых символов в тексте указанного поля происходит сравнение следующих букв и т. д. Если поле числовое, то сортировка производится в порядке возрастания (или убывания, если вы это указали) числовых величин, содержащихся в этом поле.

Индекс. Следует отметить, что сортировка файла — процесс довольно длительный. Если файл содержит тысячи записей, то на его обработку могут потребоваться десятки минут. Поэтому часто используют иной способ упорядочения — индексирование. Для файла данных создается указатель или индекс, который заносится в отдельный **индексный файл.** В индексном файле хранятся не сами записи, а лишь их номера, причем в такой последовательности, как если бы они были отсортированы по указанному полю. Порядок записей в исходном файле при этом не меняется, поэтому процедура индексирования выполняется во много раз быстрее, чем сортиров-

ка. У индексирования есть и еще одно преимущество. Иногда в работе с базой данных удобно использовать копии файла, упорядоченные по разным полям. При обычной сортировке придется отсортировать исходный файл столько раз, сколько его копий создается. Кроме того, каждая копия занимает в памяти столько же места, сколько и сам файл, а при работе с большими базами данных это очень неудобно. При индексировании исходный файл остается неизменным, а место в памяти занимают лишь индексные файлы, имеющие значительно меньший объем, так как в них хранятся только номера записей. Если вы выберете из меню функцию «индекс», система предложит вам задать поле, по которому будет выполняться операция индексирования, и ввести имя создаваемого индексного файла. Это имя, как и всякое имя файла в dBASE, может включать в себя до восьми символов. К заданному вами имени автоматически будет добавлено расширение .NDX. Теперь напомним, что когда мы говорили о поиске нужной записи, то отмечали, что команда «найти» обрабатывает только индексированные файлы. Заметим, что поиск по команде «найти» осуществляется значительно быстрее, чем по команде «установить».

Копировать. Как следует из названия, эта команда предназначена для создания копий файла данных. Такая процедура может понадобиться, например, в том случае, когда вы хотите внести в файл какие-то изменения, но при этом сохранить и прежний его вариант. Выберите функцию «копировать», нажмите «ввод», и программа предложит вам задать имя новой копии рабочего файла, а затем скопирует этот файл и запишет его на диск под новым именем.

Упаковать. Как уже упоминалось, команда «удалить» не стирает физические записи, а только помечает те из них, которые следует уничтожить. По команде «упаковать» происходит перезапись файла с удалением всех помеченных записей.

Утилиты. Утилитами называются специальные обслуживающие программы операционной системы. Утилиты dBASEIII позволяют, не прерывая выполнения программы, выполнять некоторые операции, которые обычно выполняются в среде операционной системы.

Установить файл. В отличие от команды «открыть файл» с помощью этой функции можно выбрать файл любого типа, а не только файл данных.

Копировать файл. Эта команда позволяет скопировать не только файл данных, как в режиме «организации», но и любой другой, записанный на вашем диске.

Каталог. Эта функция позволяет вывести на экран список всех хранящихся на диске файлов.

Переименовать. Как ясно из названия, эта функция позволяет изменить имена файлов.

Удалить. В отличие от аналогичной команды режима «изменение» позволяет удалить целиком весь файл, причем файл любого типа.

Изменить структуру. Как бы тщательно вы ни продумывали структуру файлов своей базы данных, рано или поздно наступит момент, когда вам захочется внести в нее изменения. Например, добавить новые поля или уничтожить те, которые больше не нужны. Может также возникнуть необходимость изменить длину какого-либо поля или тип хранящихся в нем данных. В этом случае вам на помощь придет команда «изменить структуру». Конечно, можно создать новый файл, с новой структурой, а затем переслать в него данные из старого, но это гораздо сложнее. О том, как использовать эту команду, мы расскажем ниже.

Теперь рассмотрим более подробно работу с основными описанными функциями.

Создание файла данных. Если вы впервые начинаете работать с dBASEIII, то прежде всего нужно создать какой-либо файл данных. Сначала определите, какова будет его структура, т. е. из каких полей будет состоять каждая запись этого файла. Напомним, что поле имеет имя, заданную длину и характеризуется типом данных, которые предполагается в нем хранить.

Итак, для начала следует с помощью клавиш-стрелок переместить яркое окно на экране дисплея в то место меню, где находится слово «создать». После нажатия клавиши «ввод» на экране появится надпись: «Введите имя файла». Наберите на клавиатуре какое-нибудь имя (оно не должно иметь более восьми букв), например КАДРЫ. После нажатия клавиши «ввод» система выдаст следующее предложение:

Введите структуру записи:
ПОЛЕ ИМЯ, ТИП, ДЛИНА, ДЕСЯТ
001

Курсор на экране расположится под словом «ИМЯ». Введите имя поля (оно может содержать до десяти символов). Наберите, например, ФИО. Поставьте запятую и задайте тип данных, которые будут храниться в этом поле, — С (символы). Далее опять через запятую поставьте число символов, определяющее длину поля, например 20. В последней графе записывается число десятичных знаков после запятой у числовых полей. Так как в данном случае поле символьное, эту графу заполнять не надо. Нажмите «ввод». Курсор переместится на вторую строку, помеченную цифрой 2. Теперь можно задать характеристики второго поля и т. д. После того как выпишите все необходимые поля, которые хотите иметь в файле КАДРЫ, нажмите клавишу «ввод», не заполняя очередную строчку. На экране появится сообщение: «Ввод данных? (Да/Нет)». Это означает, что создание структуры файла завершено и вы можете либо сразу начать вводить в этот файл данные, либо, нажав клавишу Н, отложить процедуру ввода на будущее. Теперь, если нажата Н, то произойдет возврат к основному меню. Для того чтобы приступить к вводу данных, нажмите клавишу Д. На экране появится примерно такая картинка:

Запись 001
ФИО : :
ОТДЕЛ : :
ДОЛЖНОСТЬ : :
ОКЛАД : :

В первом столбце стоят имена заданных вами полей. Двоеточия определяют границы каждого поля. Теперь можно просто впечатать в каждое поле необходимую информацию. Для перехода к следующему полю нажимайте «ввод». Если поле заполнится целиком, то курсор автоматически перескочит к началу следующего. Обратите внимание на то, что если тип поля определен как числовой (например, поля ОКЛАД), а вы попробуете вводить в него буквы, то у вас ничего не получится. Курсор останется на месте и буквы на экране не появятся. После того как будет завершен ввод информации в последнее поле, на экране высветится точно такая же форма для ввода данных в следующую запись. Если вы хотите закончить ввод данных, то просто нажмите клавишу «ввод».

Формирование отчета. После того как в вашем файле накопится некоторое количество записей, можно сформировать отчет по содержимому базы данных. Если вы только что загрузили программу dBASE, то прежде всего следует открыть файл, из которого вы хотите выбирать данные для отчета. Для этого используется функция «открыть файл». Если нужный файл уже открыт, то можно сразу выбрать из основного меню функцию «создать отчет» и нажать клавишу «ввод». На экране появится список полей рабочего файла с указанием длины и типа каждого поля. Под ним будет напечатано: «Введите заголовок отчета». В нижней части экрана располагаются характеристики отчета:

Ширина страницы (симв)	80
Поле слева (симв)	8
Поле справа (симв)	0
Строк /странице	58
Двойной интервал? (Д/Н)	Н

Вы можете ввести с клавиатуры текст заголовка, который будет печататься в начале каждой страницы отчета. Можно также изменить размеры полей страницы, число символов в строке или заказать печать через два интервала. При этом следует учесть, что если отчет будет выводиться не на принтер, а на экран дисплея, то не следует задавать число символов в строке больше 80, так как иначе строка не поместится на экране. Если вы не хотите менять предлагаемые системой парамет-

ры страницы, то просто нажимайте клавишу «ввод», пока курсор переходит с одного параметра на другой. После того как будет введен последний параметр, на экране появится новая подсказка, предлагающая описать разновидность отчета (при этом в верхней части экрана по-прежнему будет виден справочный список полей файла):

Группа/подгруппа по:
Только суммарный отчет? (Д/Н) :Н
Прогон после подгруппы? (Д/Н) :Н
Заголовок группы/подгруппы:
Группа/подгруппа по:
Заголовок группы/подгруппы:

Что здесь подразумевается под группой и подгруппой? Дело в том, что отчет можно разбить на группы по признаку какого-либо поля. Так, например, если в файле КАДРЫ поле ОТДЕЛ содержит номера нескольких отделов, то в каждую группу отчета будут собраны сотрудники лишь одного какого-то отдела. Однако это разбиение можно выполнять лишь по такому полю, по которому файл отсортирован или индексирован. Итак, в первую строку подсказки нужно ввести имя ключевого поля:

Группа/подгруппа по: ОТДЕЛ

В следующей строке ставится символ Д, если отчет печатается целиком, без разбиения на группы. Если поставить букву Д в третьей строке, то при печати отчета после каждой группы будет происходить автоматический прогон бумаги к началу следующей страницы. Очередная строка предлагает ввести заголовок каждой группы. Но этим возможности системы не исчерпываются.

dBASEIII допускает еще один уровень разбиения. Например, группы, объединенные по признаку ОТДЕЛ, можно разбить на подгруппы ЛАБОРАТОРИЯ (если такое поле имеется в вашем файле данных). Если разбиения на подгруппы не требуется, то две последние строки подсказки нужно пропустить, нажимая клавишу «ввод».

Следующий шаг в формировании отчета — описание столбцов. В верхней части экрана по-прежнему записана сводная информация о структуре файла, а под ней располагается такая подсказка:

Поле: дес. знаков:0 Сумма? (Д/Н): Н

	1
заголовок	2
столбца	3
	4
ширина	1

Прежде всего нужно определить имя поля, из которого будут выбираться данные в первый столбец отчета. Наберите на клавиатуре ФИО. Если бы заданное поле было числового типа, то нужно бы было задать число десятичных разрядов после запятой, т. е. определить точность числа. Кроме того, можно было бы потребовать, чтобы программа подсчитала сумму всех чисел в столбце. Для символьного поля это не нужно, и курсор автоматически переместится в строку для определения заголовка столбца. В строке «ширина» цифра 1 автоматически изменится на 20 (заданная в файле длина поля ФИО). Вы можете задать произвольный заголовок и оставить или изменить ширину столбца. Если задать ее большей, чем длина соответствующего поля, то при печати отчета между первым и вторым столбцами будут поставлены пробелы. Если же указать ширину меньшую, чем длина поля ФИО, то часть символов при печати может не поместиться в столбец и будет отброшена. После того как вы в последний раз нажмете «ввод», на экране появится форма для определения параметров второго столбца и т. д. Если вы не хотите формировать очередной столбец, просто нажмите клавишу «ввод», и создание отчетной формы завершится.

Таким образом, весь процесс формирования отчета происходит по подсказкам системы. Кроме описанных, на экран выводится еще одна подсказка:

Поле 2 осталось позиций: 52

Эта строка позволяет следить за тем, сколько свободного места остается в отчетной форме для размещения следующих столбцов.

Аналогичным образом можно формировать почтовые наклейки. Для того чтобы вывести на экран дисплея или на принтер готовый отчет, нужно открыть соответствующий файл данных, затем с помощью клавиш-стрелок перейти в режим «вывод» и выбрать в нем функцию «отчет». На запрос системы «Введите имя отчета» следует выбрать из представленного на экране списка отчетных форм ту, которая нужна в данный момент, и нажать клавишу «ввод». Все записи файла будут распечатываться в том порядке, который задан в этой форме.

Редактирование записей. Для корректировки содержимого базы данных используется функция «редактор» в режиме «изменение». При обращении к этой функции на экране высвечивается надпись «Введите номер записи». Если нужная вам запись находится где-то в середине файла и вы не знаете ее номер, можно предварительно отыскать то, что вам требуется с помощью функций поиска в режиме «перемещение». После ввода номера записи на экране появится примерно такой текст:

Запись 003
ФИО: Волокитин А. С. :
ОТДЕЛ: 121 :
ДОЛЖНОСТЬ: Нач. отд. :
ОКЛАД: 250 :

Таким образом, вы увидите содержимое всех полей нужной записи. Перемещая курсор по экрану, можно вставлять в текст отдельные символы, удалять части текста или переписывать его заново. Кроме того, можно переходить к предыдущей или последующей записи файла. Для ускорения работы с редактором можно использовать специальные клавиши. Как правило, это комбинации из двух клавиш. Так, например, чтобы перейти к работе со следующей записью файла, нужно нажать клавишу CTRL и, удерживая ее в нажатом положении, нажать клавишу с буквой С. Для обозначения такой комбинации мы будем использовать запись ⟨CTRL/C⟩. При работе в режиме редактирования используются следующие клавиши и их комбинации:

⟨CTRL/C⟩ или ⟨PGDN⟩ — переход к следующей записи;
⟨CTRL/R⟩ или PGUP — возврат к предшествующей записи;
⟨CTRL/N⟩ — вставка пустой записи;
⟨CTRL/T⟩ — удаление записи.

Для того чтобы вставить в текст один или несколько символов, нужно перейти в режим «вставка», нажав клавишу ⟨INS⟩. Во время работы в этом режиме в правом верхнем углу экрана появляется надпись «ВСТАВКА». При этом все, что вы вводите с клавиатуры, вставляется в то место строки, в котором находится курсор, а остальные символы смещаются вправо. Выход из режима «вставка» осуществляется повторным нажатием клавиши ⟨INS⟩. Для удаления символа, под которым находится курсор, следует нажать клавишу ⟨DEL⟩. Заканчивается редактирование каждого поля нажатием клавиши «ввод», после чего курсор на экране перемещается к началу следующего поля. Если нажать клавишу «ввод», когда курсор находится в последнем поле записи, то на экран высветится следующая запись, а все сделанные изменения будут занесены в файл. Сохранить внесенные изменения можно, нажав клавиши ⟨CTRL/W⟩, при этом текущая запись будет записана в файл в измененном виде.

Все это может показаться несколько утомительным, но мы ни в коем случае не предлагаем читателю запоминать все эти комбинации клавиш. Важно понять логику действий при работе с базой данных. Она в целом остается такой же и при работе с другими СУБД. Но если у вас под рукой имеется компьютер, то весьма полезно, да и увлекательно, проверить, как работают описанные команды. При этом вы сможете убедиться, насколько быстро запоминаются наиболее употребимые из них. Первые навыки вырабатываются буквально через несколько минут.

Изменение структуры файла. Возможность изменения структуры файлов придает программе dBASE дополнительную гибкость, которая зачастую необходима при работе с базами данных, так как все мы живые люди,

и рано или поздно у нас появляется потребность усовершенствовать то, что было сделано раньше, каким бы хорошим оно ни казалось. Для внесения необходимых изменений нужно в основном меню перевести выделенное окно (оно ярче других) на функцию «изменить структуру» и, естественно, нажать клавишу «ввод». Перед этим, конечно, нужно не забыть открыть файл, который вы хотите модифицировать. На экране появится такой текст:

С: КАДРЫ. DBF

	Имя поля	Тип	Длина	Дес.
1	ФИО	C	20	
2	ОТДЕЛ	N	4	
3	ДОЛЖНОСТЬ	C	15	
4	ОКЛАД	N	4	

Курсор располагается под буквой Ф в слове ФИО. Для внесения изменений в этот текст можно использовать те же клавиши, что и при редактировании файла данных. Так, если вы нажмете ⟨CTRL/N⟩, то между записями 1 и 2 появится новая пустая строка:

С: КАДРЫ. DBF

	Имя поля	Тип	Длина	Дес.
1	ФИО	C	20	
2				
3	ОТДЕЛ	N	4	
4	ДОЛЖНОСТЬ	C	15	
5	ОКЛАД	N	4	

Можно снова удалить эту строку, нажав клавиши ⟨CTRL/T⟩. А можно определить новое поле, записав в соответствующие столбцы его имя, тип и длину. Клавиши-стрелки позволяют перемещать курсор в любое место экрана и вносить изменения в нужных строках и столбцах. После того как процесс преобразования завершен, нажмите клавиши ⟨CTRL/END⟩. При этом все записи старого файла автоматически будут переписаны в новый файл. Если вы добавили в структуру новое поле, то оно, естественно, останется пустым. Обмен информацией происходит только между полями, имеющими одинаковые имена. Поэтому при изменении структуры следует соблюдать осторожность: если вы просто измените имя поля, то информация из поля со старым именем в него переписана не будет! Частичная потеря данных может произойти и при изменении длины поля. Так что, как говорил Козьма Прутков: «Бди!»

Формирование условий. Во многих перечисленных выше командах можно задавать условия отбора нужных записей файла. Мы уже упоминали об этом при описании команды поиска. Но кроме простейших условий вроде ФИО=«В», можно строить и более сложные и использовать их не только в командах поиска, но и в других, таких, как «подсчет», «сумма», «отчет» и т. д. Условия отбора записей задаются с помощью выражений отношения (=, ⟨, ⟩, ⟩=, ⟨= и ⟨ ⟩), т. е. (равно, меньше, больше, больше или равно меньше или равно и не равно). Эти отношения объединяются логическими связками (И, ИЛИ, НЕ). Логические связки принято записывать латинскими буквами, выделяя их точками: (.AND., .OR., .NOT.). С помощью выражений отношения можно сравнивать не только числовые, но и символьные величины. Каждому символу компьютер приписывает некоторое числовое значение (код) и сравниваются фактически эти коды. Коды букв обычно увеличиваются в алфавитном порядке, и в этом смысле справедливы выражения а⟨б, р⟩в и т. д. Сравнение символьных строк выполняется слева направо. Если первые символы двух строк совпадают, то сравниваются следующие до тех пор, пока не будет обнаружено несовпадение или достигнут конец строки. В результате проверки выражению присваивается значение ИСТИННО или ЛОЖНО. Так, выражение а⟩б является истинным, а выражение аа⟩аб — ложным. При последовательной проверке записей выбираются те из них, для которых заданное условие оказывается истинным.

Более сложные выражения строятся с помощью логических связок по следующим правилам. Выражение А .AND. В истинно тогда и только

тогда, когда и A и B являются истинными. A .OR .B принимает значение ИСТИННО, если истинно либо A, либо B. Связка .NOT. меняет значение выражения с ИСТИННО на ЛОЖНО, и наоборот. Для того чтобы определить последовательность сравнения выражений, их можно группировать, используя скобки. Так, например, по условию ФИО=«В» .AND. (.NOT. ОТДЕЛ=121) будут отобраны только записи, содержащие данные о сотрудниках всех отделов, кроме 121, фамилии которых начинаются с буквы В. Отметим еще раз, что, записывая условия, содержимое символьных полей следует выделять кавычками (в нашем примере это буква В), а содержимое числовых полей — писать без кавычек (номер отдела). При работе с основным меню условия задаются по подсказке. Так, например, если вы выберете из меню функцию «подсчет», то на экране появятся записи «выполнить команду» и «условие поиска». Первая запись выделена на экране повышенной яркостью. Если нажать клавишу «ввод», то программа подсчитает общее количество записей открытого в текущий момент файла. Если же с помощью клавиши-стрелки перейти к строке «условие поиска», то после нажатия клавиши «ввод» на экране появится подсказка, предлагающая ввести имя поля и записать формулировку условия для анализа содержимого этого поля. Такая же подсказка появляется и при выборе из меню функции «отчет». Дело в том, что когда вы формировали отчет, то компьютер записал в память только его общие характеристики: число и заголовки столбцов, их ширину, имена полей и т. д. Но при выводе отчета на экран или на принтер можно с помощью дополнительных условий отбирать из всего массива записей лишь те, которые требуются в данный момент.

У читателя может возникнуть законный вопрос: «А где же раздобыть столь замечательный пакет?» У нас в стране пользователи персональных ЭВМ пока в основном переписывают такие программы друг у друга. Но в Москве и в некоторых других городах уже созданы кооперативы программистов, в которых можно не только записать необходимые программы, но и получить подробные инструкции по их использованию, а при необходимости и приспособить их под конкретную задачу.

Командный файл

На хорошем языке программирования можно записать все что угодно, даже протокол профсоюзного собрания.

Из разговора программистов

До сих пор мы рассматривали только один режим использования dBASE III — режим ASSIST. И хотя работа с экранным меню позволяет выполнять некоторые последовательности команд (например, сначала отобрать нужные записи файла данных, а затем вывести их на экран), программирование как таковое здесь не использовалось. Конечно, многим пользователям персональных компьютеров достаточно возможностей, предоставляемых режимом ASSIST, для создания и эксплуатации собственных баз данных, но рано или поздно может возникнуть естественный вопрос: «А какие дополнительные удобства сулит более глубокое изучение системы?» Ведь в конце концов плохо не брать от программы всего того, что она может дать. В этом разделе мы рассмотрим некоторые дополнительные формы работы с dBASE.

СУБД dBASE III имеет свой собственный язык команд, на котором можно записать все основные операции, необходимые для работы с базой данных, в том числе, конечно, и все операции, предлагаемые в основном меню режима ASSIST. Команды языка dBASE III записываются латинскими буквами и используют английскую мнемонику, т. е. являются сокращениями соответствующих слов английского языка. Но так как число этих команд сравнительно невелико, то их нетрудно запомнить и тем, кто не владеет английским. Кроме того, пользователь в любой момент может обратиться за помощью к системе,

послав команду HELP (помощь) или просто нажав клавишу с обозначением F1. По этой команде на экран выводятся имена всех команд с кратким разъяснением способов их употребления. На клавиатуре персональных компьютеров имеются так называемые «функциональные клавиши» с обозначениями F1, F2, ..., F10. Нажатие каждой такой клавиши обычно заменяет какую-либо часто встречающуюся команду. Большинство систем используют клавишу F1 для вызова функции HELP.

Для того чтобы перейти от режима работы по меню к командному режиму, нужно нажать клавишу ⟨ESC⟩. Экран при этом очистится и в его левом нижнем углу появится мигающая точка — «приглашающий символ», означающий, что система готова к приему команды. Чтобы вновь вернуться к работе по основному меню, достаточно нажать функциональную клавишу F2.

После появления на экране приглашающего символа можно вводить любую команду. Причем если при вводе была допущена ошибка, например простая опечатка, то ничего страшного не произойдет. Система не «узнает» команду и выведет на экран сообщение: «Неизвестная команда. Нужна помощь? (Да/Нет)». Если ответом будет «Да», то на экран будет выведен список команд. При ответе «Нет» на экране снова появится приглашающий символ. Так что различного рода сообщения и подсказки используются и при работе в командном режиме.

Приведем еще один пример. Для того чтобы открыть файл данных, не прибегая к основному меню, нужно послать команду USE (использовать) и указать имя файла. Если вы не укажете имя файла, на экране появится надпись: «Введите имя файла». Если вы укажете имя уже открытого файла, система предупредит вас соответствующим сообщением. Аналогичным образом система поддерживает выполнение других команд.

Мы не будем здесь разбирать все команды dBASE. Их описание можно при желании найти в специальных руководствах или вывести на экран командой HELP. Перечислим лишь самые необходимые и те, которые не входят в функции основного меню. Команды можно посылать системе по одной, набирая их на клавиатуре. Но обычно команды объединяются в группы, образуя программу. Последовательность команд, записанная в память компьютера, образует командный файл. Имена командных файлов в dBASE III имеют расширение .PRG.

Командные файлы можно создавать, редактировать, записывать в память компьютера и уничтожать, если в них отпадает необходимость. Чтобы создать или отредактировать командный файл, нужно послать команду MODIFY COMMAND (набрать ее на клавиатуре и нажать клавишу «ввод»). Система попросит ввести имя файла, с которым вы желаете работать (до восьми символов), и автоматически добавит к нему расширение PRG. Если вы укажете имя уже существующего файла, то на экране появится текст записанной в нем программы, и в него можно внести любые изменения. Если же имя файла встречается впервые, dBASE очистит экран и выведет на его сообщение «новый файл». На пустом экране можно записывать команды, нажимая после каждой клавишу «ввод» и набирая следующую команду с очередной строки. Для перехода с одной записи на другую можно использовать те же клавиши, что и при редактировании файла данных. После того как формирование командного файла закончено, его нужно сохранить, записав в память компьютера клавишами ⟨CTRL/W⟩. Программа будет записана в память под указанным именем. Теперь, для того чтобы выполнить всю последовательность команд, записанных в этом файле, достаточно набрать на клавиатуре слов DO (выполнить) и имя нужного файла.

В этом заключается первое преимущество использования командных файлов. Они позволяют автоматически выполнять целую последовательность команд. Так, для того чтобы в режиме ASSIST найти нужную запись файла данных, вам может понадобить-

ся выполнить такую последовательность операций:

1. Выбрать функцию «установить дисковод» и указать имя дисковода, в котором хранится диск, содержащий базу данных.

2. Выбрать функцию «открыть файл» и указать имя файла данных.

3. Выбрать функцию «найти» и задать условие поиска нужной записи.

4. Выбрать функцию «продолжить» для поиска следующей записи, удовлетворяющей заданному условию и т. д.

Конечно, это не очень обременительно, если всю последовательность операций нужно выполнить один раз. А если в течение дня приходится прибегать к этой процедуре десятки раз? Здесь использование командного файла может оказаться очень удобным, так как для его запуска достаточно набрать на клавиатуре DO имя файла.

Рассмотрим, какие команды могут понадобиться для создания такого файла.

USE имя файла. Эта команда открывает файл данных и обычно именно с нее начинаются командные файлы.

CLEAR. Очищает экран, т. е. удаляет с него всю выведенную раньше информацию.

LOCATE. Отыскивает нужную запись по заданному условию.

FOR. Определяет условие поиска.

CONTINUE. Команда продолжения поиска следующей записи, удовлетворяющей условию.

DISPLAY. Выводит на экран найденную запись, на которую установлен указатель.

Используя эти команды, можно написать небольшую программу поиска нужной записи. Для начала нужно набрать на клавиатуре:

MODIFY COMMAND ПОИСК

Когда экран очистится, можно приступить к вводу команд (не забывая после каждой строки нажимать клавишу «ввод»).

```
USE КАДРЫ
CLEAR
LOCATE FOR ФИО=«В» .AND. ОТДЕЛ=121
DISPLAY
CONTINUE
DISPLAY
```

Затем нужно сохранить файл, нажав ⟨CTRL/W⟩. Если теперь просмотреть каталог файлов, хранящихся на диске, вызвав из основного меню функцию «каталог», то в списке окажется новый файл с именем ПОИСК. PRG. Чтобы инициировать его выполнение, достаточно набрать на клавиатуре

DO ПОИСК

Этот файл можно запускать сколько угодно раз, и по его командам на экран будут выводиться две первые записи файла КАДРЫ, содержащие информацию о сотрудниках отдела 121, фамилии которых начинаются с буквы В.

Описанная программа имеет ряд очевидных недостатков. Во-первых, она отыскивает только две первые нужные записи, а писать команды вывода информации на экран и продолжения поиска неудобно. Для того чтобы автоматизировать этот процесс, можно воспользоваться командой, организующей цикл, которая имеет такой вид:

```
DO WHILE условие продолжения
команды...
ENDDO
```

Все команды, заключенные между предложениями DO WHILE (выполнять пока) и ENDDO (закончить выполнение), будут обрабатываться снова и снова, до тех пор пока условие продолжения цикла остается ИСТИННЫМ. Как только оно примет значение ЛОЖНО, выполнение цикла завершится. Условие продолжения можно записать так же, как задавались условия поиска записей, сравнивая значение какого-либо поля с константой. Но обычно требуется просмотреть файл целиком, проверив все его записи. В этом случае можно использовать специальную функцию «конец файла» — EOF ( ). Она принимает значение ЛОЖНО, когда указатель установлен на любую запись файла, и ИСТИННО, когда достигается конец файла.

Еще одна команда, используемая в большинстве программ, — это

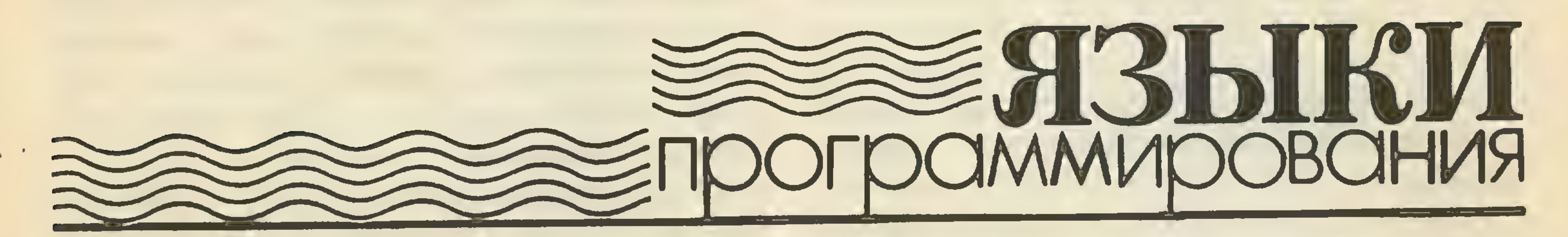
SKIP — переход к следующей записи. Она позволяет переводить указатель на несколько записей вперед (например, SKIP 3 — перейти вперед на три записи) или назад (SKIP — 3 — вернуться на три записи назад).

Но если файл данных содержит много записей, удовлетворяющих условию поиска, то они будут очень быстро сменять друг друга на экране дисплея и пользователь просто не успеет прочесть выводимую информацию. Чтобы приостановить выполнение программы, можно использовать команду WAIT (ожидать). Возобновится работа лишь после того, как пользователь нажмет какую-нибудь (любую) клавишу.

Кроме того, программу можно упростить. Использование команды LOCATE для поиска нужной записи в данном случае не является необходимым. Для вывода на экран содержимого записи, удовлетворяющей условию поиска, можно вставить это условие прямо в команду DISPLAY. Тогда командный файл ПОИСК можно переписать в таком виде:

```
USE КАДРЫ
CLEAR
DO WHILE .NOT. EOF( )
DISPLAV FOR ФИО=«В» .AND. ОТ-
ДЕЛ=121
  SKIP
   WAIT
ENDDO
```

Теперь записи будут появляться на экране по одной, поиск следующей записи будет начинаться после нажатия любой клавиши, и программа не



М. П. МАЛЫХИНА,
А. М. ЧАСТИКОВ

Имя языка составлено из первых букв названия COBOL — COmmon Business—Oriented Language, что дословно означает «общий язык деловой ориентации». Язык этот ориентирован на обработку экономической информации. Он разработан рабочей группой, созданной под эгидой исполнительного комитета по языкам систем обработки данных Кодасил (CODASYL—Conference on Data System Languages).

Работы над первой версией языка завершены в декабре 1959 г., а в феврале 1960 г. опубликован предварительный отчет. На разработку КОБОЛ-60 потребовалось около 4 человеко-лет. Корни КОБОЛа уходят к ранним, малоизвестным языкам программирования ФЛОУМАТИК, КОМТРАН и ФАКТ, а АЛГОЛ повлиял на выбор символов для КОБОЛа. Первые компиляторы КОБОЛа созданы в декабре 1960 г. одновременно двумя фирмами RCA и Remington — Rand — Univas.

При разработке КОБОЛа ставились цели — сделать язык машинозависимым и приблизить его к естественному языку, с тем чтобы для непрофессионального программиста программы были читаемыми. По утверждению многих пользователей, с этой точки зрения КОБОЛ, возможно, самый лучший в отличие от других языков программирования, имеющих формальный синтаксис. В КОБОЛе взят синтаксис английского предложения, поэтому-то программы на этом языке легко читать.

КОБОЛ был первым языком, в котором средства описания данных соответствуют процедурным возможностям, и первым языком, в котором введен тип данных «запись», являющийся основной структурой данных. К одной из примечательных особенностей КОБОЛа относится рекурсивное описание данных. Другая его особенность в том, что программы на КОБОЛе разбиваются на части, называемые разделами, причем каждая программа состоит из четырех разделов: идентификации, оборудования (среды), данных и процедур.

Раздел идентификации служит для установления тождественности программы и содержит различные пояснения, необходимые для ее документирования. Раздел оборудования содержит данные об используемом оборудовании, в основном периферийном. Раздел данных содержит информацию о типе и структуре данных, организации и распределении памяти и т. п. Раздел процедур содержит алгоритмы вычислений. В свою очередь, эти основные части программы разбиваются на более простые: секцию, параграф, предложение и слово.

Еще одна отличительная черта КОБОЛа — это его постоянное изменение и совершенствование. Вслед за появлением КОБОЛа-60 в следующем году была опубликована вторая версия под названием «КОБОЛ-61», которая получила широкое распространение, однако имела некоторую несовместимость с КОБОЛом-60. В

закончит работу, пока не будут просмотрены все записи файла.

Однако и в таком виде наш командный файл далек от совершенства. Поставим перед собой следующие задачи:

— обеспечить вывод информации на экран порциями по 10 записей;

— выводить на экран содержимое только поля ФИО из каждой записи;

— снабдить информацию на экране пояснительным текстом.

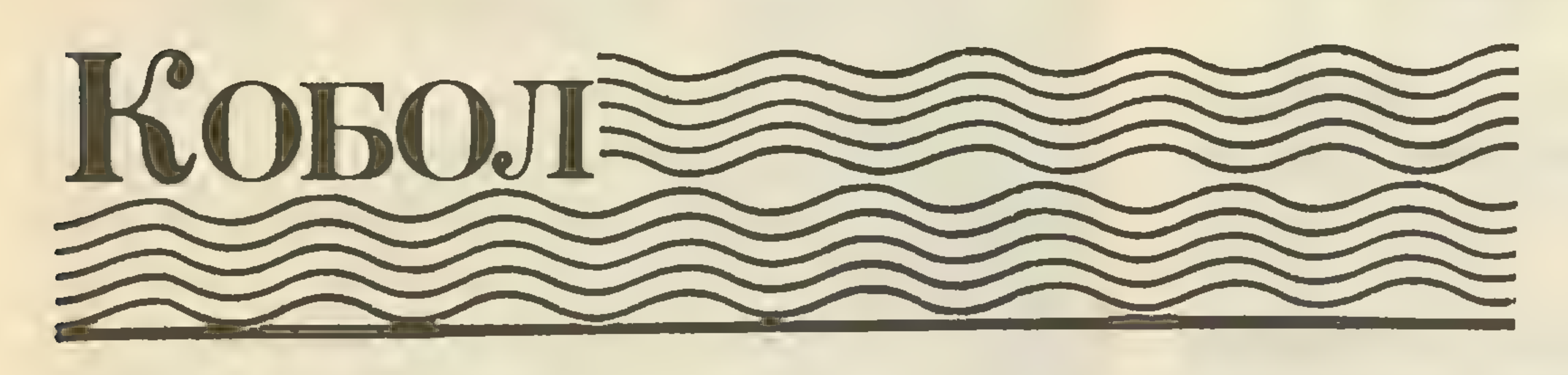
Для решения поставленных задач нам понадобятся всего три новые команды и одно новое понятие. Это понятие переменных, без использования которых не обходится практически ни одна программа. Переменные нужны для временного хранения каких-либо единиц или элементов данных. Обычно информация, записанная в переменную, используется только во время работы программы. После выключения компьютера, а иногда и после завершения выполнения программы информация, записанная в переменные, удаляется из памяти. В отличие от многих других языков программирования в dBASE III переменная создается тогда, когда ей присваивается какое-либо значение. Одновременно определяется ее тип (символьный, числовой и т. д.). Для присвоения переменной значения можно использовать одну из двух эквивалентных форм записи:

```
STORE 2.326 TO A
```

или

```
A=2.326
```

И в том и в другом случае в памяти



1963 г. была опубликована расширенная версия языка, названная «Расширенный КОБОЛ-61». Два года спустя появилась новая версия с несколько необычным именем: «КОБОЛ, редакция 1965». Эта версия в качестве американского национального стандарта утверждена в 1968 г. Однако работы по усовершенствованию языка и выработке новых версий продолжаются.

Измененный американский стандарт КОБОЛа принят в 1974 г. с соответствующим названием — КОБОЛ-74. В настоящее время в Американском национальном институте стандартов (ANSI) заканчивается разработка другого стандарта КОБОЛа, который предусматривает введение ряда новых конструкций в язык и отказ от некоторых редко используемых или неудобных операторов и т. д. Новый стандарт предусматривает также значительное сокращение различных подмножеств языка.

В СССР первые компиляторы с подмножества языка КОБОЛ реализованы в 1968 г. на ЭВМ «Днепр-21» и «Минск-32», а в 1977 г. был принят отечественный стандарт на язык программирования КОБОЛ (ГОСТ 22558—77).

Оценивая вклад этого языка в теорию и практику программирования, нельзя не указать на противоречивый характер отношений к нему пользователей, с одной стороны, и специалистов по информатике, с другой. Если среди программистов он получил распространение благодаря своей удобочитаемости, а может быть, и ранней стандартизации, то многие ученые его появление восприняли как ошибку, а его использование как «болезнь», с которой необходимо бороться. Причем некоторые из ученых, в частности известный голландский специалист по программированию Э. Дейкстра, выражали свое негативное отношение к КОБОЛу в довольно резкой форме (ACM SIGPLAN NOTICE, 1982, v. 17, pp. 13—15). Они возражали против использования английского языка как основы КОБОЛа из-за его несовершенной стилистики. Хорошая читаемость программ, утверждали они, не говорит в пользу КОБОЛа, так как программы с введением многочисленных «шумовых» слов становятся слишком многословными. Из-за большой длины программы компиляторы работают медленно, а кроме того, возникают трудности автоматического обнаружения ошибок во время компиляции.

В заключение отметим: несмотря на то что опыт разработки и применения КОБОЛа, очевидно, мало повлиял на создание языков, появившихся после него (за исключением языка ПЛ/1), все же надо признать, что он оставил заметный след в истории развития языков программирования.

Литература

Ющенко Е. Л. и др. КОБОЛ (Программированное учебное пособие). — К.: Наукова думка, 1973.

Быкова В. П. и др. КОБОЛ ЕС ЭВМ. — М.: Статистика, 1978.

Пратт Т. Языки программирования. Разработка и реализация. — М.: Мир, 1979.

Хротко Г. М. Языки программирования высокого уровня для микро-ЭВМ. — М.: Междунар. НИИ проблем управления, 1985.

Хелмс Г. Л. Языки программирования / Пер. с англ. — М.: Радио и связь, 1985.

компьютера образуется переменная числового типа с именем А. Под этим именем будет храниться число 2.326 (в программировании вместо запятой в десятичных дробях используется точка). Значение числа может меняться, на то она и переменная, но его всегда можно узнать, обратившись к переменной по имени. В сущности, переменная отличается от поля тем, что она не связана ни с каким файлом данных, и тем, что она уничтожается после окончания работы. Если команду присвоения записать в виде

B=«Имя сотрудника»

то в памяти компьютера образуется переменная символьного типа с именем В, хранящая символьную строку «Имя сотрудника». Числовые переменные можно использовать в обычных арифметических выражениях:

C=A+1

В этом случае переменной С присваивается значение, равное 3.236. Справедлива и такая запись:

A=A+1

Она приводит к увеличению на единицу значения переменной А. Заметим, что в языке dBASE, как и во многих других языках программирования (а их число превышает 2000!), действие умножения обозначается звездочкой (*), а деления — косой чертой (/). Обратим внимание на то, что при использовании переменных приходится следить за тем, чтобы не смешивать в одном выражении переменные разных типов. Так, например, запись A=B приведет к ошибке, так как мы определили А и В соответственно как числовую и символьную переменные.

Следующая полезная команда — это знак вопроса (?). Она выводит на экран информацию, записанную следом. Так, например, по команде

? «Сумма чисел А и С = »,A+C

на экран будет выведен текст, заключенный в кавычки, и результат сложения переменных А и С:

Сумма чисел А и С = 5.762

Еще одним мощным средством языка dBASE является команда, позволяющая выполнять какие-либо действия при истинности заданного условия. Это команда IF ...ELSE... ENDIF. Если заданное условие оказывается истинным, то выполняются команды, записанные после IF (если). В противном случае управление передается командам, следующим за ELSE (иначе). В некоторых случаях слово ELSE можно опустить. Тогда при ложности условия все команды, записанные внутри условного оператора, опускаются. Заканчивается список входящих в этот оператор команд словом ENDIF.

Используя вышеописанные команды, программу ПОИСК можно переписать в таком виде:

```
USE КАДРЫ
CLEAR
N=1
N1=0
DO WHILE .NOT.  EOF ( )
N1=N1+1
?N, «Фамилия сотрудника», ФИО
N=N+1
IF N1=10
N1=0
WAIT
ENDIF
SKIP
ENDDO
```

Если запустить эту программу командой DO ПОИСК, то на экран будут выведены десять строк текста, каждая из которых состоит из порядкового номера, фразы «Фамилия сотрудника» и содержимого поля ФИО очередной записи. После того как пользователь просмотрит текст, он может нажать любую клавишу, и на экран будут выведены следующие десять записей. Если нужно выводить записи не подряд, а выборочно, то это нетрудно сделать, введя еще один оператор условия.

В данном примере мы ввели две переменные. Переменная N служит для подсчета числа записей. Она увеличивает на единицу при каждом повторении цикла и выводится на экран в начале каждой строки. Переменная N 1 служит для управления остановом программы. Ее значение увеличивается на единицу до тех пор, пока не станет равным десяти, после чего выполнение команд прерывается, и счет начинается сначала.

Обратите внимание на то, что часть команд программы записана со сдвигом от левого края страницы. Это сделано для удобства чтения текста программы. Команды можно записывать в любом месте строки.

Следующей командой, создающей пользователю дополнительные удобства, является команда форматированного вывода. С ее помощью можно размещать выводимые данные в произвольном месте экрана. Это позволяет формировать отчеты произвольной формы, в дополнение к отчетам, состоящим из столбцов, создаваемых из основного меню. Команда форматированного вывода в общем виде записывается так:

```
@ N1, N2 SAY A
```

Здесь N1 и N2 — числа или числовые переменные, задающие соответственно строку и столбец, в которых будет находиться начало выводимой информации. У большинства современных персональных компьютеров на экране дисплея помещается 24 строки по 80 символов в каждой. Поэтому экран можно представить в виде таблицы из 24 строк и 80 столбцов. Порядковые номера строк и столбцов этой таблицы и задаются числами N1 и N2. Величина А может быть именем переменной, либо именем поля. Можно вместо А подставить и просто текст сообщения, при этом не забыв заключить его в кавычки. Используя форматированный вывод, можно переписать командный файл ПОИСК в таком виде:

```
USE КАДРЫ
CLEAR
N=1
N1=0
DO WHILE .NOT. EOF (   )
   N1=N1+1
   @ N, S SAY N
   @ N, 8 SAY «Фамилия сотрудника»
   @ N,27 SAY ФИО
   N=N+1
   IF N1=10
          N1=0
            WAIT
   ENDIF
   SKIP
ENDDO
```

Здесь переменная N использована не только для нумерации записей, но и для вывода каждой записи в следующей строке экрана. На экран в каждой строке выводятся: переменная, текст сообщения и содержимое поля ФИО. Начинается вывод каждой из этих величин с 5-го, 8-го и 27-го столбца соответственно.

Команда форматированного вывода, конечно, создает определенные удобства, но для организации настоящего взаимодействия или диалога с программой необходимо иметь возможность вводить данные с клавиатуры и присваивать их значения переменным или записывать в поля файлов данных. Такую возможность предоставляет пользователю команда форматированного ввода. В общем виде она записывается так:

```
@ N1, N2 GET A
READ
```

Здесь, так же как и в команде форматированного вывода, N1 и N2 — числа, задающие строку и столбец экрана, а А — имя поля или переменной. Естественно, что перед тем как быть использованной в команде форматированного вывода, переменная должна быть определена командой присвоения. А перед использованием поля должен быть открыт соответствующий файл данных. Рассмотрим простейший пример.

```
A=«          »
@ 10,15GET A
READ
```

Если вставить такую последовательность команд в программу, то по первой команде будет создана символьная переменная с именем А и в нее будут записаны десять пробелов. По команде форматированного ввода с 15-го столбца в 10-й строке содержимое этой переменной будет выведено на экран. Так как в переменную записаны только пробелы, то в указанном месте экрана появится светящийся отрезок строки длиной в десять символов, не содержащий никакого текста. Выполнение программы будет приостановлено в ожидании ввода. Набрав на клавиатуре нужный текст, следует

нажать клавишу «ввода», после чего этот текст будет записан в переменную, и выполнение программы возобновится. Если заполнить символами все десять позиций, отведенных переменной A, то ввод произойдет автоматически. Клавишу «ввод» нажимать не потребуется. С помощью команд форматированного ввода и вывода можно организовать диалог с компьютером по заранее разработанному сценарию. Для выбора различных режимов работы можно создавать свои собственные меню, например такого же типа, как основное меню режима ASSIST.

Хотя мы рассмотрели здесь лишь небольшое число команд, они позволяют выполнять почти все основные манипуляции с данными. Большинство остальных команд либо дублируют команды режима ASSIST, либо позволяют более рационально и экономно организовать процесс обработки. Кроме того, еще несколько команд, которых мы не касались, предназначены для одновременной обработки нескольких файлов данных, в частности, для выполнения операций пересылки, слияния и т. п. Это лишний раз подтверждает возможность постепенного изучения dBASE и углубленного освоения его средств в процессе активной работы с базами данных.

Теперь скажем несколько слов об отличиях различных версий dBASE. В отличие от dBASE III более ранняя версия — dBASE II может функционировать не только под управлением операционной системы MS—DOS, но и в среде операционной системы CP/M. Это позволяет устанавливать ее, например, на таких компьютерах, как Роботрон 1715 производства ГДР. Ограничения, накладываемые на основные параметры файлов, у dBASE II более жесткие, чем у dBASE III. Так, число полей одного файла данных в dBASE II не может превышать 32. Общее число символов в каждой записи не должно быть больше 1000. Максимально допустимый размер файла данных составляет 65 536 записей. Кроме того, у dBASE II нет режима ASSIST и все команды требуется набирать на клавиатуре в явном виде, не прибегая к помощи меню. В dBASE III изменена форма записи некоторых команд по сравнению с dBASE II, но эти изменения очень незначительны. Увеличены в dBASE III и возможности выполнения вычислений: введены функции возведения в степень, вычисления логарифмов и некоторые другие.

Более поздняя версия — dBASE III+ отличается от dBASE III прежде всего тем, что позволяет производить обмен данными между компьютерами, объединенными в сеть. Другие отличия в основном сводятся к улучшению сервиса, т. е. созданию дополнительных удобств для пользователя.

Заключение

Упомянем здесь еще об одной появившейся сравнительно недавно форме организации диалога человека с компьютером. Речь идет о взаимодействии с компьютером при помощи наглядных картинок, пиктограмм, или, как их еще называют, «икон». Основным инструментом при такой работе служит так называемая «мышь», представляющая собой небольшую коробочку, соединенную с компьютером, которую можно катать по поверхности стола. Передвижение «мыши» по столу отображается на экране дисплея соответствующим перемещением маленькой стрелки-указателя, которую легко подвести к любому месту экрана. Пиктограммы на экране изображают предметы, располагающиеся обычно на рабочем столе. Это — записная книжка, дневник, телефонный справочник, папки для бумаг, калькулятор и т. п. Подведя стрелку-указатель к нужному предмету и нажав кнопку «мыши», пользователь вызывает на экран увеличенное изображение выбранной области. Если это, например, записная книжка, то можно перелистать ее страницы (на экране при этом видна динамическая картинка переворачивающихся страниц), сделать запись, стереть или перечеркнуть ее. Если надобность в каком-то документе отпала, то можно подвести стрелку к изо-

бражению мусорной корзины и «выбросить» этот документ. Создание базы данных при такой форме работы превращается в «перекладывание» карточек, изображающих на экране поля, из которых составляются записи. Как гласит старинная китайская поговорка: «Одна картинка может сказать больше, чем 10 000 слов». И действительно, такая наглядная форма работы значительно упрощает и ускоряет процесс взаимодействия с данными. Немаловажно и то, что пользователю не приходится перестраиваться психологически. Он все время имеет дело с привычными предметами, которые постоянно окружают его в повседневной жизни.

Каждый год на рынке программного обеспечения появляются сотни и тысячи новых программ. Многие из них подаются покупателю под девизом: «Получайте максимум возможностей, затрачивая минимум времени на работу и изучение». Не исключено, что уже завтра будут созданы программы, использующие какие-то совершенно новые принципы работы с базами данных и анализа заложенной в них информации. А осмысленный анализ (формализация) накопленной информации лежит в основе многих (если не всех) научных открытий. Достаточно вспомнить знаменитый периодический закон Менделеева. Наверное, можно утверждать, что, располагай Менделеев персональным компьютером с современной системой управления базами данных, ему было бы гораздо легче нащупать закономерность, наличие которой он интуитивно чувствовал. В доказательство можно привести совсем свежий пример. В 1987 г. в «Докладах АН СССР» появилась статья с сообщением о попытке систематизации свойств химических соединений, предпринятой сотрудниками Куйбышевского политехнического института им. Куйбышева под руководством академика В. В. Кафарова. В память компьютера были занесены сведения о различных характеристиках химических соединений. Всего учитывалось около 10 000 веществ. Если расположить эти соединения в порядке возрастания суммы атомных номеров составляющих их химических элементов, то вряд ли удастся выявить какую-либо закономерность в их свойствах. Ведь свойства молекул характеризуются и молекулярными массами, и могут зависеть от плотности, отражающей взаимное расположение атомов в молекулах. С помощью ЭВМ исследователям удалось проанализировать зависимость физико-химических свойств веществ от перечисленных признаков. В результате была установлена определенная закономерность. Оказалось, что вещества с одинаковыми суммами атомных номеров, молекулярных масс и плотностями обладают весьма схожими физико-химическими свойствами. А это позволяет прогнозировать свойства вещества на основе данных о его составе и плотности! Вряд ли такой анализ можно бы было выполнить, не прибегая к помощи современной компьютерной техники.

На прощание пожелаем читателю, чтобы эти и многие другие примеры убедили его в необходимости как можно скорее приобщиться к увлекательному миру СУБД и ЭВМ и чтобы это принесло ему много радостных минут и неожиданных открытий.

Список рекомендуемой литературы

Дейт К. Введение в системы баз данных.— М.: Мир, 1980.

Мартин Дж. Организация баз данных в вычислительных системах.— М.: Мир, 1980.

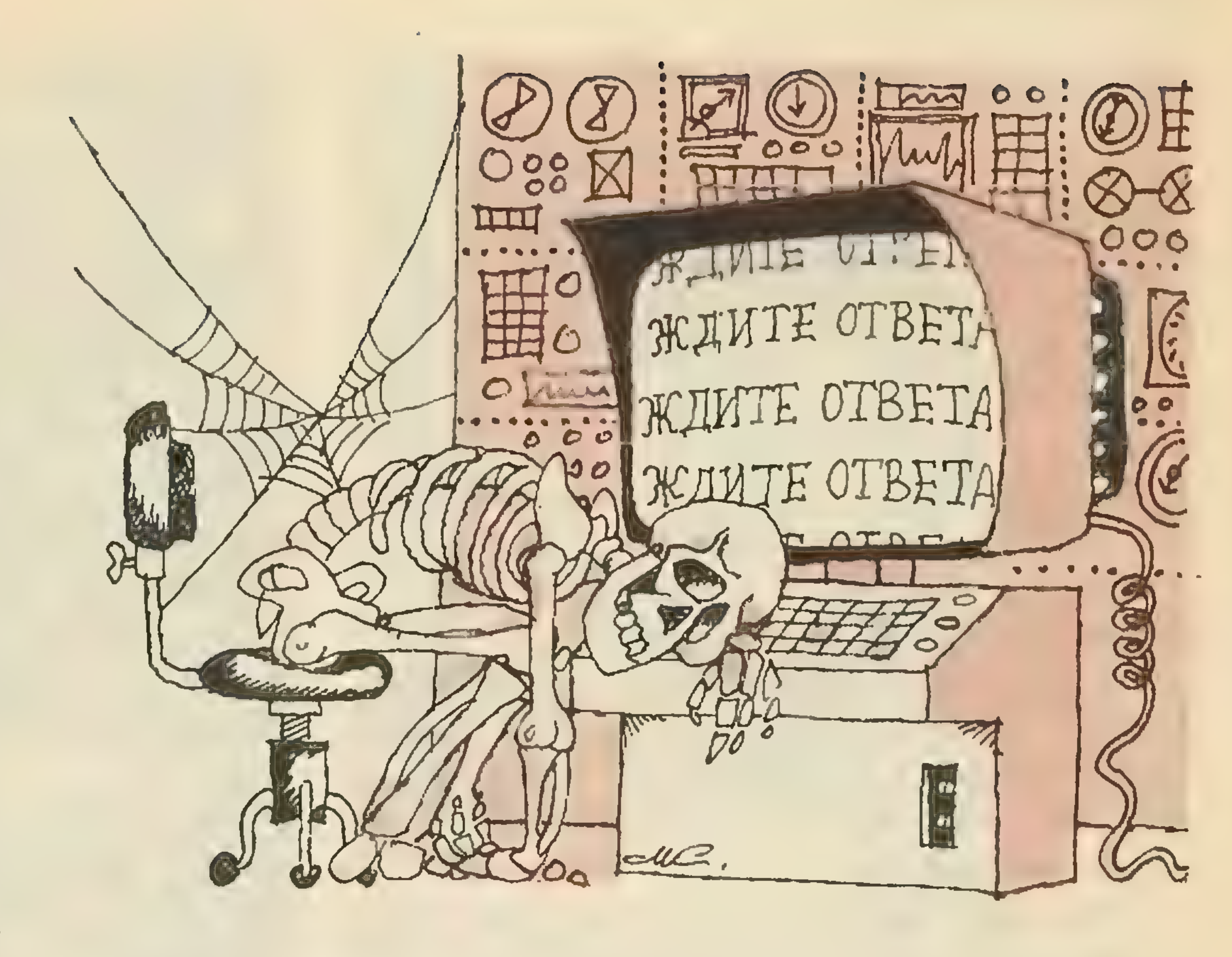
Тиори Т., Фрай Дж. Проектирование структур баз данных — М.: Мир, 1985.

Крамм Р. dBASE II и dBASE III — системы управления базами данных для персональных ЭВМ. — М.: Финансы и статистика, 1988.

«ЭЛЕКТРОННЫЙ СМОГ»

Паразитные излучения в атмосферу теперь образно называют «электронным смогом» (см. «New Seientist», Англия, том 118, № 1607, 7 апреля 1988 г., стр. 34—38).

При экспериментах с двумя маломощными радиопередатчиками, проведенными компанией «Би-Би-Си» близ Шекспировского театра в Стратфорде-на-Эвоне (для проверки возможности строительства радиостанции с шестью мощными радиопередатчиками), произошло нарушение машинных программ управления осветительным оборудованием театра, а также стирание информации в трех электронных пишущих машинках.



Специалисты «Электрикал рисерч ассошиэйшн текнолоджи», имеющей заказ от министерства обороны на изучение проблем помехозащиты, установили, что уровни излучений, создаваемые большинством бытовых ЭВМ, значительно превышают нормы, предусмотренные стандартом «BS 65276» в диапазоне частот 30—200 МГц.

Одним из серьезных примеров воздействия помех является гибель в 1984 г. в ФРГ истребителя «Торнадо», причиной которой оказался его пролет вблизи мощных радиопередатчиков широковещательных радиостанций. Специалисты считают, что излучения этих радиопередатчиков вызвали сбои в ЭВМ системы управления полетом самолета.

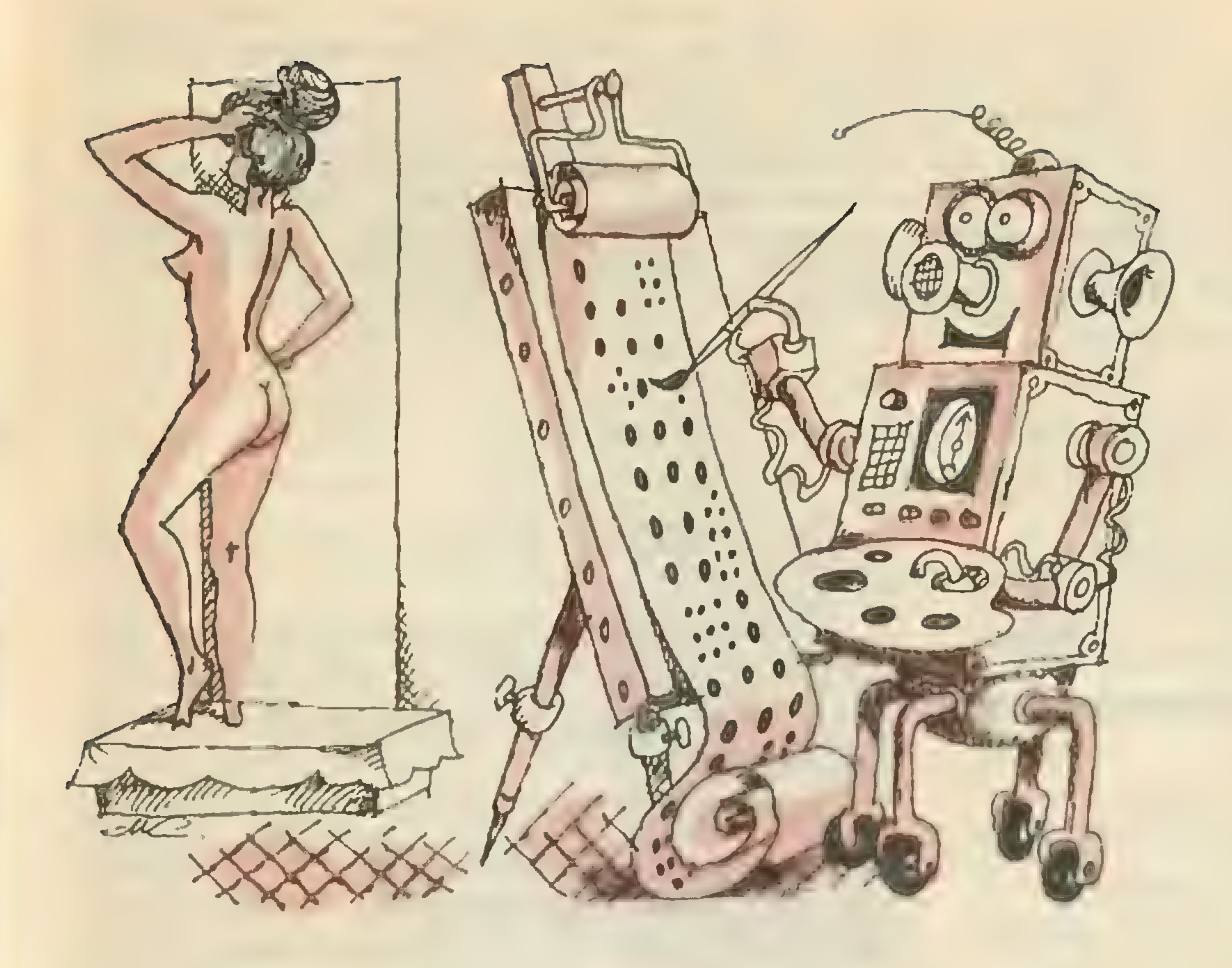
Имеются также рекомендации в стандарте «BS 65276» на ограничения уровней излучения ЭВМ и автоматизированной учрежденческой технологии, при соблюдении которых дальность пеленгации радиоизлучений бытовых и учрежденческих ЭВМ не будет превышать соответственно 10 и 30 м. Нормы на допустимые электрические помехи уже введены в законодательном порядке с 1984 г. почтовыми ведомствами США и ФРГ.

Специалисты почтового ведомства Голландии провели исследования возможностей перехвата сигналов, излучаемых видеоиндикаторной аппаратурой ЭВМ, в которой происходит усиление слабых импульсов до сотен тысяч вольт. Кроме возможного вредного воздействия на живой организм, такие импульсы, с учетом современной техники восстановления пропадающих импульсных сигналов, могут быть зарегистрированы на расстояниях до нескольких сотен метров.



В заключение о результатах наблюдений, особо актуальных в эпоху массового использования средств вычислительной техники. Конторские служащие и обслуживающий дисплеи персонал все чаще жалуются на ощущения раздражения лица и глаз. Эти симптомы были признаны проблемой производственной гигиены (1978 г.). Позже эту причину отнесли к климатическим факторам.

Из последних результатов исследований все эти симптомы связывают с высокими электрическими полями. Было выяснено, что напряженность электрического поля



возрастает по мере приближения к лицу, особенно к области носа и лба. В противоположность возрастанию силы электрического поля скорость воздуха по мере приближения к лицу падает.

Выяснено также, что микроорганизмы в электрических полях также переносятся — это показала группа финских исследователей.

В сильных электрических полях, излучаемых экранами дисплеев, переносятся мелкие частицы, микроорганизмы, аллергены, минеральные волокна. В непосредственной близости от лица напряженность электрического поля обеспечивает их перенос и осаждение. Небольшая скорость воздушного обмена и высокое содержание загрязнений усиливают влияние электрического поля и значение электроклимата (см.: Prophylace und Ergonomie. 1985, 35, № 12, 378).

...Выяснилось, что дисплей отнюдь не столь безобиден, как телевизор. Из каждых десяти человек, работающих на ЭВМ, девять жаловались на неприятные ощущения, и семеро из них имели на то объективные причины: исследования установили, что у них ухудшается зрение, растет утомляемость нервной системы, учащаются аллергические заболевания. Был найден и виновник — низкокачественные мониторы компьютеров, не обеспечивающие достаточно резкого, яркого и контрастного изображения. Даже обычная частота «мельканий» экрана оказалась здесь слишком малой: телезритель, наблюдающий подвижное изображение, их почти не замечает, а оператор ЭВМ, работающий в более напряженных условиях, испытывает от этого добавочное утомление. Особенно сильно могут пострадать дети. Во-первых, они куда больше, чем взрослые, увлечены новой чудесной игрушкой, во-вторых, мозг ребенка гораздо чувствительней к мельканию изображения.

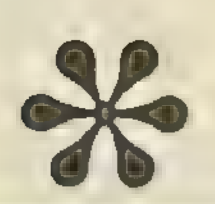
Поэтому в 1985 г. специальная группа экспертов Всемирной организации здравоохранения опубликовала рекомендации по более строгому контролю качества компьютерных мониторов. Большинство западных стран ужесточило соответствующие гигиенические нормы, а профсоюзы стали требовать от работодателей их неукоснительного соблюдения. Отныне крупнейшие мировые производители ЭВМ выпускают технику, удовлетворяющую рекомендациям ВОЗ (см.: Литературная газета. — 1988. — 27 января).





Если открыть «Перечень программных средств, поставляемых минским опытным заводом программно-технических средств и систем в 1988 году» (Минское НПО «Центрисистем»), то на первой странице в разделе ПС общего назначения под пунктом 1.6 можно найти название СУБД РЕПЕР, которая является «системой управления базами данных реляционного типа для персональных профессиональных ЭВМ», типа ЕС-1840, ЕС-1841, работающих в операционной системе Альфа-ДОС (или MS-DOS). Что же это за система!

марки
ТИПЫ
характеристики



СУБД РЕПЕР

ДЛЯ ОТЕЧЕСТВЕННЫХ ПЕРСОНАЛЬНЫХ ЭВМ

Ю. Е. ВАДЯСКИН,
С. А. ФРОЛОВ

Мы только что познакомились с одной из лучших в мире СУБД dBASE III, предназначенной для использования на персональных ЭВМ фирмы IBM и на совместимых с ними компьютерах. В том, что dBASE III является действительно одним из лучших продуктов на рынке программных средств общего назначения для ПЭВМ, сомневаться не приходится. Это СУБД в жесткой конкурентной борьбе с другими аналогичными системами обеспечила себе несомненный приоритет. Сегодня для ПЭВМ фирмы IBM на СУБД dBASE III и ее предшественницу dBASE II приходится более половины установок среди общего числа реляционных систем. А это означает, что система работает на сотнях тысячах машин и приносит разработчикам многомиллионные прибыли.

Следует ли из этого, что она должна работать на всех отечественных машинах? Может быть, сегодня, когда мы хорошо знаем о заметном отставании СССР от западных стран в области персональных компьютеров, нам и не нужно других СУБД? Давайте приобретем эту систему, лучшую в мире, и дело решено — отечественные ПЭВМ получат необходимую программную «начинку» в сфере реляционных СУБД?!

Однако даже самое поверхностное рассмотрение такого «очевидного» варианта ставит трудноразрешимые вопросы.

Во-первых, dBASE III является англоязычной системой. Другими словами, она ориентирована на людей (профессионалов и непрофессиональных пользователей), владеющих английским языком.

Может быть, попробуем перевести ее на русский? Увы, это не так просто. Дело в том, что фирма поставляет систему, записанную на гибких магнитных дисках (ГМД) только в загрузочных кодах. Разобраться в устройстве программ, представленных в загрузочных кодах, и что-то там изменить, это все равно, что пытаться что-то исправить в электронных схемах компьютера, не имея документации. Задача, конечно, принципиально выполнимая, но стоимость ее решения может свести экономический результат почти к нулю.

Кроме того, подобные изменения могут внести в программы новые ошибки, искажения, и разобраться, кто виноват — фирма-поставщик или автор ее модификации (адаптации), будет очень трудно.

Развивать, т. е. строить новые версии СУБД, расширять ее возможности будет и вовсе невозможно. За каждой новой включенной в систему функцией придется обращаться в фирму. Следует также принять во внимание и то, что фирма, заботясь о своих прибылях, продает систему с различными ограничениями: на число копий, которые можно снять с фирменных ГМД; на число инсталляций (т. е. установок на ПЭВМ) системы; без права продажи копий. А чтобы подтвердить и под-

крепить свои авторские права, юридические меры защиты дополняются техническими и программными.

Отсюда ясно, что полная ориентация на зарубежную систему, особенно если речь идет о ПС общего назначения, рассчитанного на многотысячное тиражирование и применение непрофессиональными пользователями, едва ли оправдана. Необходимо создавать свои аналогичные системы, авторами которых являются наши соотечественники. Эти СУБД должны быть сразу ориентированы на отечественные машины и вобрать в себя опыт и свойства своих зарубежных аналогов. Подобные СУБД уже создаются.

Система РЕляционного типа для ПЕРсональных ЭВМ (РЕПЕР) по своим функциональным возможностям в основном аналогична dBASE III и ориентирована на ПЭВМ Единой системы или программно-совместные с ними модели. РЕПЕР относится к оригинальным отечественным разработкам, в которой с самого зарождения предусмотрено двуязычное общение (диалог) с пользователем на русском и английском языках. Хотя в данном случае для разработчиков, в руках которых есть исходные тексты, а не загрузочные модули, не принципиально, на каком языке спроектирован диалог: русском, польском, английском или даже китайском.

РЕПЕР может служить базовой СУБД для организации различных систем обработки информации и информационно-поисковых систем на основе ПЭВМ (например, АСУ промышленных предприятий и непромышленной сферы, систем автоматизированного проектирования — САПР, автоматизированных рабочих мест, информационно-справочных систем, систем индивидуального делопроизводства). Система представляет собой комплекс программных средств, обеспечивающих автоматизацию создания, ведения и поиска данных табличной структуры, относящихся к произвольной предметной области. Она ориентирована на пользователей, не являющихся профессионалами-программистами. СУБД может использоваться также административно-техническими работниками и служащими различных категорий, а также профессиональными разработчиками информационных систем (прикладными программистами).

Минимальная конфигурация ПЭВМ (ЕС-1841), достаточная для использования СУБД РЕПЕР, включает следующие устройства:

— процессор с оперативной памятью 384 Кбайт;

— клавиатуру;

— цветной графический дисплей;

— печатающее устройство (принтер);

— один накопитель на гибком магнитном диске (ГМД);

— один накопитель на магнитном диске типа «Винчестер».

С некоторыми ограничениями (без использования цветовых и графических возможностей) система может работать и на первых моделях (ЕС-1840 с алфавитно-цифровым дисплеем и двумя накопителями на ГМД без «Винчестера»). Пользователю в этом случае поставляется специальная версия СУБД РЕПЕР.

Система РЕПЕР предназначена для работы в операционной системе Альфа-ДОС (либо с аналогичной ОС фирмы IBM MS-DOS). Ведутся работы по переносу СУБД и на другие отечественные компьютеры. Успешно работает система на ПЭВМ «Искра-1030». Наличие исходных текстов программ делает ее мобильной и позволяет в кратчайшие сроки перенести ее и на машины «Нейрон», «Прайвец», СМ-1810 и др. Следует заметить, что СУБД РЕПЕР прекрасно себя чувствует и на зарубежных ПЭВМ, прежде всего фирмы IBM, что открывает возможность ее поставок в другие страны.

Быстрый доступ к данным осуществляется на основе метода индексирования файлов. В системе поддерживается одновременная обработка информации из нескольких (до 10) файлов. РЕПЕР поддерживает данные реляционного (табличного) типа фиксированной структуры, состоящей из записей (строк) и полей (столбцов). Поля имеют фиксированный формат и содержат данные: символьные, числовые

(целые и дробные), логические (со значениями «истина» и «ложь»), даты в форме ДД/ММ/ГГ, текстовые длиной до 4 Кбайт.

Текстовые поля хранятся в виде отдельных файлов и при обращении к ним подключаются к файлам базы данных. Даты можно использовать в арифметических операциях сложения и вычитания. Поле записи может быть объявлено индексным ключом и использоваться в качестве критерия поиска информации. Для хранения промежуточных данных используются оперативные переменные, размещаемые в основной памяти или сохраняемые во внешней памяти.

Взаимодействие пользователя с СУБД РЕПЕР осуществляется в диалоговом однопрограммном режиме. К важной особенности системы относится наличие единого командного языка высокого уровня, построенного на основе русской лексики в формате ключевых слов. Командный язык РЕПЕРа

— обеспечивает описание данных и манипулирование ими, также выполнение всех функций системы;

— охватывает оба режима функционирования (диалоговый и пакетный);

— является универсальным и замкнутым (не требует выхода в традиционные языки программирования);

— строится в формате ключевых слов и обладает развитым синтаксисом;

— относится к процедурным и включает обработку отдельных записей;

— обеспечивает обработку сложных арифметических и булевых выражений;

— является реляционно-полным и относится к типу исчисления с переменными — кортежами.

Каждая команда имеет вид фразы, состоящей из имени команды (глагола) и следующих за ним вспомогательных слов и параметров, определяющих вариант исполнения.

СУБД РЕПЕР разрешает ввод команды после появления на экране дисплея знака равенства «=». Часть команд системы представляет меню для выбора или специально оформленный экран для ввода данных.

Всего в языке системы насчитывается 65 команд. В состав языка входит специальная группа команд программирования, которые:

— выполняются только в пакетном режиме;

— обеспечивают создание структурированных программ;

— строятся на основе английской или русской лексики; включают возможность организации циклов проверки условий, создания подпрограмм, оказания ввода данных с клавиатуры и т. п.

В совокупности языковых средств СУБД РЕПЕР позволяет создавать сложные прикладные программные системы обработки данных.

Выполнение программ, созданных средствами системы РЕПЕР, осуществляется в режиме интерпретации. Для отладки таких программ в РЕПЕРе предусмотрены специальные отладочные средства.

Диалоговые команды языка ориентированы преимущественно на пользователей-непрограммистов, хотя могут быть широко использованы и в программах. Для удобства работы в языке отсутствуют зарезервированные слова, а имена и ключевые слова команд разрешается сокращать до трех букв (произвольно меняя их окончания).

Среди диалоговых команд выделяется группа полноэкранных команд, которые представляют сеанс диалога с выдачей подсказки и специальной видеограммы. В окнах этой видеограммы пользователь вводит описания, данные или корректуры. К таким командам относятся: СОЗДАТЬ ФАЙЛ, ДОБАВИТЬ, РЕДАКТИРОВАТЬ, МОДИФИЦИРОВАТЬ СТРУКТУРУ и др. Некоторые команды функционируют в режиме выбора из меню: СОЗДАТЬ ОТЧЕТ, АССИСТЕНТ, СПРАВКА.

Ряд команд манипулирования данными содержит необязательный параметр «область действия», принимающий значения ВСЕ или СЛЕДУЮЩИЕ (от текущей записи файла БД).

К этим командам относятся: КОПИРОВАТЬ, СОРТИРОВАТЬ, СУММИРОВАТЬ, ВЫДАТЬ и др. Во многих командах допускается обработка сложных условий и выражений.

Процесс удаления записей осуществляется в два этапа. Сначала записи только намечаются (логически) для удаления. Затем помеченные записи удаляются (физически) отдельной командой. До выполнения этой команды логически удаленные записи могут быть восстановлены.

В СУБД РЕПЕР поддерживаются файлы восьми типов: индексные, текстовые, форматные, командные (программы) и др. Система совместима по файлам БД и файлам программ с СУБД dBASE III, а по текстовым файлам с операционной системой Альфа-ДОС. Это позволяет после небольших доработок использовать ее программы в dBASE III. Доработки связаны с небольшими различиями в языках этих систем. Файлы БД dBASE III используются в РЕПЕР без изменений.

В составе системы имеются встроенные средства генерации программ ввода/вывода данных на основе описания видеограмм и отчетов в виде форматных файлов. Для редактирования командных, форматных и текстовых файлов имеется встроенный текстовой редактор. Для редактирования сформированных запросов имеется встроенный построечный редактор.

Дополнительную гибкость при обработке данных обеспечивает наличие 35 функций, построенных по аналогии с библиотеками подпрограмм в языках программирования. К ним относятся: ДЕНЬ, ДЛИНА, МЕСЯЦ, НОМЗАП, ПОДСТР, ВРЕМЯ, ТИП и др. Такие функции могут быть использованы в программах на языке системы, а также во встроенном в РЕПЕР калькуляторе, который позволяет вычислять сложные арифметические и логические выражения, расширяя тем самым возможность языковых средств.

В процессе работы СУБД РЕПЕР обеспечивает полноту диагностических сообщений, не требующих от пользователя обращения к дополнительным справочным материалам.

СУБД РЕПЕР разработана с использованием расширенной системы программирования ТУРБО-ПАСКАЛЬ. Основными компонентами архитектуры системы являются ядро, резидентные и оверлейные модули, текстовые файлы подсистем АССИСТЕНТ и СПРАВКА.

РЕПЕР функционирует под управлением операционной системы Альфа-ДОС, а также PC-DOS, MS-DOS.

Основные сравнительные характеристики СУБД РЕПЕР и dBASE III.

Показатели	dBASE III	РЕПЕР
Структура данных:		
максимальное число записей в файле	неогр.	неогр.
максимальное число полей в записи	128	128
максимальный размер записи, Кбайт	4	4
наличие текстового типа данных	есть	есть
Функциональные ограничения:		
максимальное число используемых оперативных переменных	256	256
количество совместно обрабатываемых файлов БД	10	10
минимальный объем требуемой ОП, Кбайт	256	256
Составитель команд типа АССИСТЕНТ	есть	есть
Генерация и печать отчетов	есть	есть
Встроенный генератор прикладных форм и программ ввода/вывода	нет	есть
Показатели надежности:		
возможность восстановления удаляемых записей	есть	есть
ведение регистрационного журнала	есть	есть
возможность общения с системой на русском языке	нет	есть
возможность модификации и развития (доступность исходных текстов)	нет	есть
Язык разработки	Си	ТУРБО-ПАСКАЛЬ

СУБД РЕПЕР может быть использована в АСУ, САПР, ИПС, АРМ раз-

личного назначения, создаваемых на предприятиях различных отраслей и в организациях непромышленной сферы, а также для автоматизации индивидуального делопроизводства.

Конечно, СУБД dBASE III, с которой мы начали разговор, как и многие программные системы, постоянно развивается, совершенствуется. На рынок поступают новые, более совершенные версии системы, она пополняется принципиально новыми возможностями, рождаются новые поколения систем. Так, dBASE превратились в dBASE II, затем в dBASE III, сейчас рынок программных средств захватывает новое поколение dBASE III+ (dBASE III плюс), предназначенное для работы в распределенных системах, в локальных вычислительных сетях.

Фирма объявила о предполагаемом выпуске новейшего продукта семейства СУБД dBASE — dBASE IV. Эта СУБД включает ряд принципиально новых возможностей в диалоговой и пакетной обработке баз данных: многопользовательский режим применения, расширения языка программирования и диалоговых средств общения, новые типы данных и т. д.

Естественно, предполагается и постоянное развитие отечественного аналога СУБД dBASE III — пакета общего назначения РЕПЕР.

Разработчики постоянно работают с ним и предполагают в первую очередь расширить РЕПЕР, включив в систему версии 2.0 следующие возможности:

— встроенные средства обработки данных в режиме электронного бланка (расчетных таблиц);

— компилятор для ускорения выполнения программ, созданных средствами языка системы;

— расширение средств машинной графики для представления информации из файлов БД;

— средства распределенной обработки информации в информационно-вычислительных сетях;

— генератор прикладных программ обработки вводимых пользователем макетов;

— библиотека проблемно-ориентированных программ на языке системы;

— обучающий курс по работе с системой;

— интерфейсы с распространенными языками программирования ПАСКАЛЬ, Си.

В настоящее время ведутся активные исследования в области создания нового поколения СУБД РЕПЕР, включающего элементы искусственного интеллекта в интерфейсе с пользователем и при оптимизации работы генератора прикладных систем. Это позволит упростить взаимодействие с системой, создавать и применять в ней не только базы данных, но также и базы знания. Основой для проведения этих перспективных работ является отечественный генератор экспертных систем «Аргумент», который также поставляется Минским НПО «Центросистем» для ПЭВМ ЕС-1841 с 1988 г.

Можно надеяться, что по своим функциональным возможностям СУБД РЕПЕР в недалеком будущем станет конкурентоспособным товаром на мировом рынке программных средств.

Хотелось бы в заключение подчеркнуть еще одну очень важную мысль. Направление СУБД dBASE развивается и совершенствуется в условиях сильной конкуренции, когда на программном рынке имеются и постоянно появляются новые разработки, впитывающие опыт предшественников и новейшие идеи в программировании. Поэтому появление перспективной отечественной системы не должно означать, что кто-то скажет: «Нам этого достаточно. Не нужно тратить зря силы на параллельные разработки». Дальнейшее развитие программных средств невозможно без их конкуренции со своими аналогами (в том числе и отечественными!), в условиях которой выживают лучшие из лучших. Появление СУБД РЕПЕР в арсенале программных средств общего назначения должно послужить вызовом для других разработчиков, стимулом для участия в работе по созданию конкурентоспособных отечественных программ.

«ТЕРМИНАЛ»
КОМПЬЮТЕРНЫЙ КЛУБ ШКОЛЬНИКОВ

Треугольники Паскаля и Лейбница

Автор: Денис Буланович — Крым, Симферопольский район, село Родниковое, 7-й класс.

Треугольником Паскаля называется числовое множество, элементы которого состоят из натуральных чисел. Треугольник Паскаля изображается так, как показано на рис. 1 а.

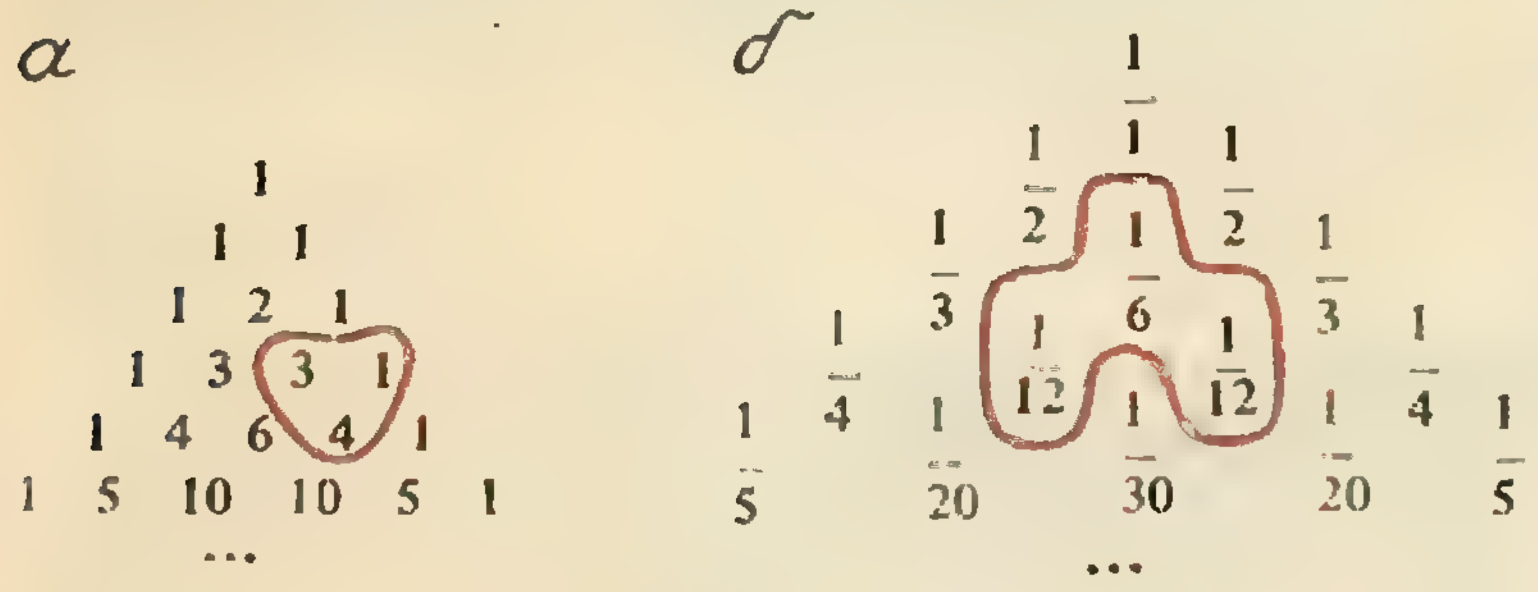


Рис. 1.

Каждый элемент треугольника Паскаля (кроме стоящей в его вершине 1) равен сумме его «северо-западного» и «северо-восточного» элементов. На рис. 1,а выделен элемент $A_{4,4}=4$, он равен сумме элементов $A_{3,3}$ и $A_{3,4}$:4=3+1. Имеет место расчетная формула:

$$A_{i,j} = A_{i-1,j-1} + A_{i-1,j}$$

Треугольником Лейбница называется множество правильных дробей вида 1/N. Треугольник Лейбница изображается так, как показано на рис. 1,б. Каждый элемент треугольника Лейбница, включая его вершину, равен сумме его «юго-западного» и «юго-восточного» элементов. На рис. 1,б выделен элемент $A_{3,2}=1/_6$, он равен сумме элементов $A_{4,2}$ и $A_{4,3}$ $1/_6=1/_{12}+1/_{12}$.

Задание на программирование

Составить программу генерирования элементов каждого из треугольников с выводом треугольников на экран (печать).

Программа должна включать элементы диалога. ЭВМ в начале работы должна запросить пользователя, какой из треугольников он намерен создавать, каким числом строк. В связи с тем что размер экрана ПЭВМ для вывода результатов ограничен, то ЭВМ по запросу пользователя должна выводить на экран величину любого элемента из каждого треугольника.

Общее описание алгоритма

Вся программа состоит из следующих крупных блоков:

— блок представления программы с элементами начального диалога;

— блок вычисления величины элементов треугольника Паскаля;

— блок вычислений величины элементов треугольника Лейбница;

— блок вывода на экран треугольника Лейбница.

```
RUN
КАКОЙ ТРЕУГОЛЬНИК СТРОИТЕ?
1. – ТРЕУГОЛЬНИК ПАСКАЛЯ
2. – ТРЕУГОЛЬНИК ЛЕЙБНИЦА
WВОД Т–ВЫБОР ТИПА ТР–КА? 1
ВВОДИТЕ КОЛ–ВО СТРОК  0<N<11 ? 7

                          1
                         1 1
                        1 2 1
                       1 3 3 1
                      1 4 6 4 1
                    1 5 10 10 5 1
OK               1 6 15 20 15 6 1

RUN
КАКОЙ ТРЕУГОЛЬНИК СТРОИТЕ?
1. – ТРЕУГОЛЬНИК ПАСКАЛЯ
2. – ТРЕУГОЛЬНИК ЛЕЙБНИЦА
WВОД Т–ВЫБОР ТИПА ТР–КА? 2
ВВОДИТЕ КОЛ–ВО СТРОК  0<N<8? 4

                          1
                          -
                         1 1
                         - -
                         2 2
                        1 1 1
                        - - -
                        3 6 3
                      1  1  1  1
                      -  --  -- -
OK                    4 12  12  4
```

Текст программы:

```
10 ' ТРЕУГОЛЬНИКИ ПАСКАЛЯ И ЛЕЙБНИЦА
20 PRINT "КАКОЙ ТРЕУГОЛЬНИК СТРОИТЕ?"
30 PRINT "1. – ТРЕУГОЛЬНИК ПАСКАЛЯ"
40 PRINT "2. – ТРЕУГОЛЬНИК ЛЕЙБНИЦА"
50 INPUT "WВОД Т–ВЫБОР ТИПА ТР–КА"; Т
60 ON T GOTO 80, 260
70 GOTO 10
80 CLS:WIDTH 80
90 INPUT "ВВОДИТЕ КОЛ–ВО СТРОК 0<N<11";N
100 IF N>10 PR N<=0 THEN PRINT "ПОВТОРИТЕ ВВОД": GOTO 90
110 DIM A(N, N)
120 PRINT
130 LET R=40
140 LET A(1, 1)=1: PRINT TAB (R) : A (1, 1)
150 FOR I=2 TO N
160 LET R=R–2
170 PRINT TAB (R);
180 FOR K=1 TO I
190 LET A (I, K)=A (I–1, K–1)+A (I–1, K)
200 IF A (I, K)< 9 THEN PRINT A (I, K);" ";:GOTO 220
210 PRINT A (I, K);
220 NEXT K
230 PRINT
240 NEXT I
250 END
260 CLS
270 INPUT "ВВОДИТЕ КОЛ–ВО СТРОК 0 < N < 8"; N
280 IF N>8 OR N<=0 THEN PRINT "ПОВТОРИТЕ ВВОД": GOTO 270
290 DIM A (N, N), B (N, N)
300 LET A(1, 1)=1: LET B (1, 1)=1
310 FOR I= 2 TO N
320 LET A (I, 1)=1:LET B (I, 1)=1
330 FOR K=2 TO I
340 LET A (I, K)=1:LET X=B (I, K–1):LET Y=B (I–1, K–1)
350 IF X=Y THEN 380
360 IF X>Y THEN LET X=X–Y ELSE LET Y=Y–X
370 GOTO 350
380 LET M=B (I, K–1): LET T=B (I–1, K–1): LET E=(M*T) /X
390 LET X=E/B (I, K–1): LET Y=E/B (I–1, K–1) : LET M=Y–X: LET B (I, K)=E/M
400 NEXT K
410 NEXT I
420 LET R=43
430 FOR I=1 TO N
440 PRINT TAB (R) ;
450 FOR K=1 TO I
460 PRINT A (I, K) " ".
470 NEXT K
480 PRINT TAB (R)
490 FOR K=1 TO I
500 IF B (I, K) <10 THEN PRINT " ";"–";" "; ELSE PRINT "—";" ";
510 NEXT K
520 PRINT TAB (R–1)
530 FOR K=1 TO I
540 IF B (I, K) <10 THEN PRINT" " ; B (I, K); ELSE PRINT B (I, K)
550 NEXT K
560 PRINT
570 LET R=R–2
580 NEXT I
590 END
```

Post scriptum

Общие выводы

Рассмотренная задача имеет для школьников общематематический познавательный характер. Если треугольник Паскаля широко известен, то треугольник Лейбница знают немногие, а эти треугольники являются своеобразными двойниками.

С точки зрения освоения техники программирования рассмотренная задача требует хорошего владения операторами PRINT и TAB и умения конструировать несложные диалоги.



НАМ ПИШУТ

«Уважаемая редакция! Пишу вам от лица формирующегося центра НТТМ, ставящего одной из своих задач компьютеризацию школы, производства, лаборатории.

Хотелось бы получить более полную информацию по описанным в ваших выпусках системам: аналоговому интегрирующему калькулятору и по процессору «Кентавр».

О. Г. Багиев,
преподаватель физики,
г. Беслан

В № 6 за 1988 г. мы рассказали об аналоговом вычислительном калькуляторе. В этом номере помещаем статью о процессоре «Кентавр».

Г. Н. Алексаков

КЕНТАВР

Проектирование новых и совершенствование действующих ядерных реакторов невозможно без тщательных экспериментальных исследований. Такие исследования ведутся на действующих реакторах и на критических стендах (сборках), представляющих собой физическую модель реактора с комплексом аппаратуры для управления и измерений.

Активная зона уран-графитового реактора представляет собой некоторый объем, заполненный графитовыми блоками (рис. 1). Блоки «нанизаны» на вертикальные трубы — «технологические каналы», расположенные на плане активной зоны в узлах равномерной сетки. В технологических каналах устанавливают кассеты с топливом — «тепловыделяющие сборки», дополнительные поглотители и регулирующие стержни. Последние можно перемещать вертикально при помощи электродвигателей, оперативно регулируя тем самым коэффициент размножения нейтронов и распределение плотности нейтронного потока по активной зоне. Тепло, выделяющееся при делении топлива и при поглощении нейтронов, отводится водой, прокачиваемой через технологические каналы. Образовавшийся пар вращает турбогенератор АЭС. Активная зона заключена в оболочку, отражающую нейтроны и препятствующую их утечке из активной зоны, а также поглощающую все остальные излучения, сопутствующие процессам в активной зоне.

Критическая сборка повторяет конструкцию реактора, но плотность нейтронного потока в ней примерно в миллион раз меньше, чем в реакторе АЭС. Для исследований этого достаточно, а уменьшение во столько же раз тепловыделения, излучений и накопления продуктов деления упрощает решение многих инженерных проблем и уменьшает стоимость экспериментальной установки. Это, однако, не дает никаких оснований для халатного отношения к таким установкам. Напротив, работа в режиме частых пусков и остановок, маневрирования полем и уровнем мощности предъявляет еще более жесткие требования на точность и надежность аппаратуры контроля, регулирования и защиты.

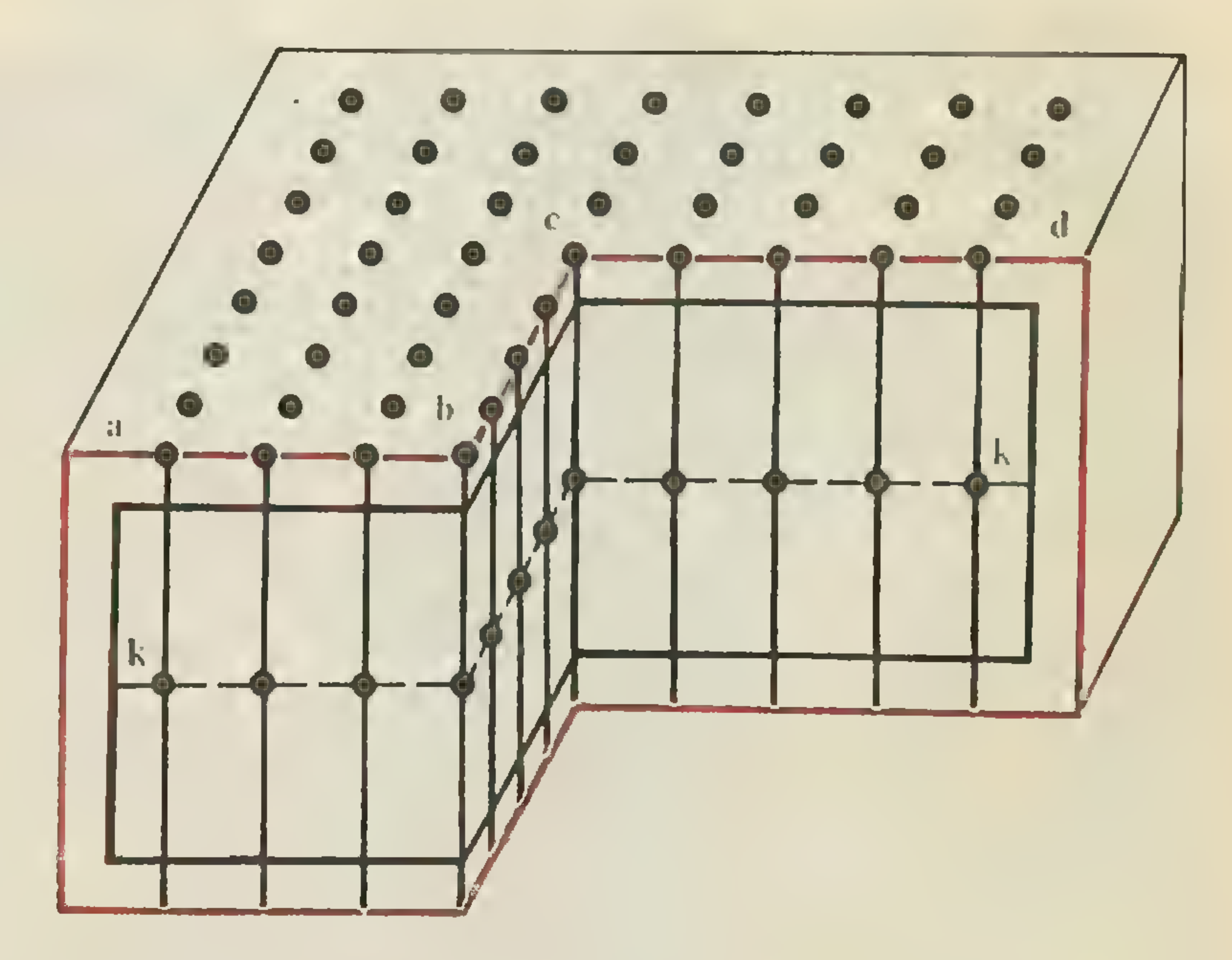


Рис. 1.

Одна из целей экспериментов на критическом стенде — изучение формы нейтронных полей для различных загрузок топлива, поглотителей и положения регулирующих стержней. Автоматизацию таких экспериментов обеспечивает система контроля и регулирования нейтронного поля «Кентавр».

Для измерения плотности нейтронного потока используют малогабаритные ионизационные камеры. Их размещают обычно в центральных трубках тепловыделяющих сборок. Выходные токи ионизационных камер

несут информацию о распределении нейтронного потока в горизонтальном сечении активной зоны. Токи камеры преобразуются в напряжения, а последние умножаются на индивидуальные коэффициенты, компенсирующие разброс характеристик и выравнивающие чувствительность всех измерительных каналов. Полученные таким образом напряжения представляют в масштабе форму нейтронного поля (рис. 2). Далее эти сигналы умножаются на коэффициенты формы поля. Величина последних выбирается такой, чтобы все выходные напряжения были одинаковы, если форма поля соответствует расчетной или заданной. Сформированные таким образом сигналы поступают далее на блок диагностики каналов и выбора управляющих стержней для компенсации отклонений получающейся формы поля от заданной.

В основе этого блока лежит пространственный резистивный фильтр ПРФ (рис. 3, а и б). С его помощью рассчитывается «сглаженное» поле с учетом всех каналов контроля для точек размещения датчиков и регулирующих стержней. «Горизонтальные» резисторы r образуют сетку с узлами, соответствующими технологическим каналам. Через «вертикальные» резисторы R к узлам D, соответствующим размещению ионизационных камер, подводятся сформированные описанным выше способом напряжения (рис. 3, а и б). Выбором отношения r/R функция отклика фильтра — распределение напряжений в узлах при подаче на один из входов напряжения, отличного от остальных (рис. 3, в), — приближена к функции отклика реактора — к распределению нормированного потока при внесении «возмущения» в соответствующий технологический канал. Поэтому напряжение в каждом из узлов ПРФ определяется совокупностью сигналов всех измерительных каналов, «взвешенных» функцией отклика ПРФ. Сравнение входных сигналов ПРФ с соответствующими выходными D позволяет выявить аномальные отклонения сигналов на выходах каналов контроля. Сравнение же выходных сигналов ПРФ в точках размещения регулирующих стержней с уставкой используется для выбора стержней, направления и величины их перемещения. Так обеспечивается наиболее эффективная компенсация отклонений формы поля от заданной.

Для представления информации о нейтронном поле и преобразованиях

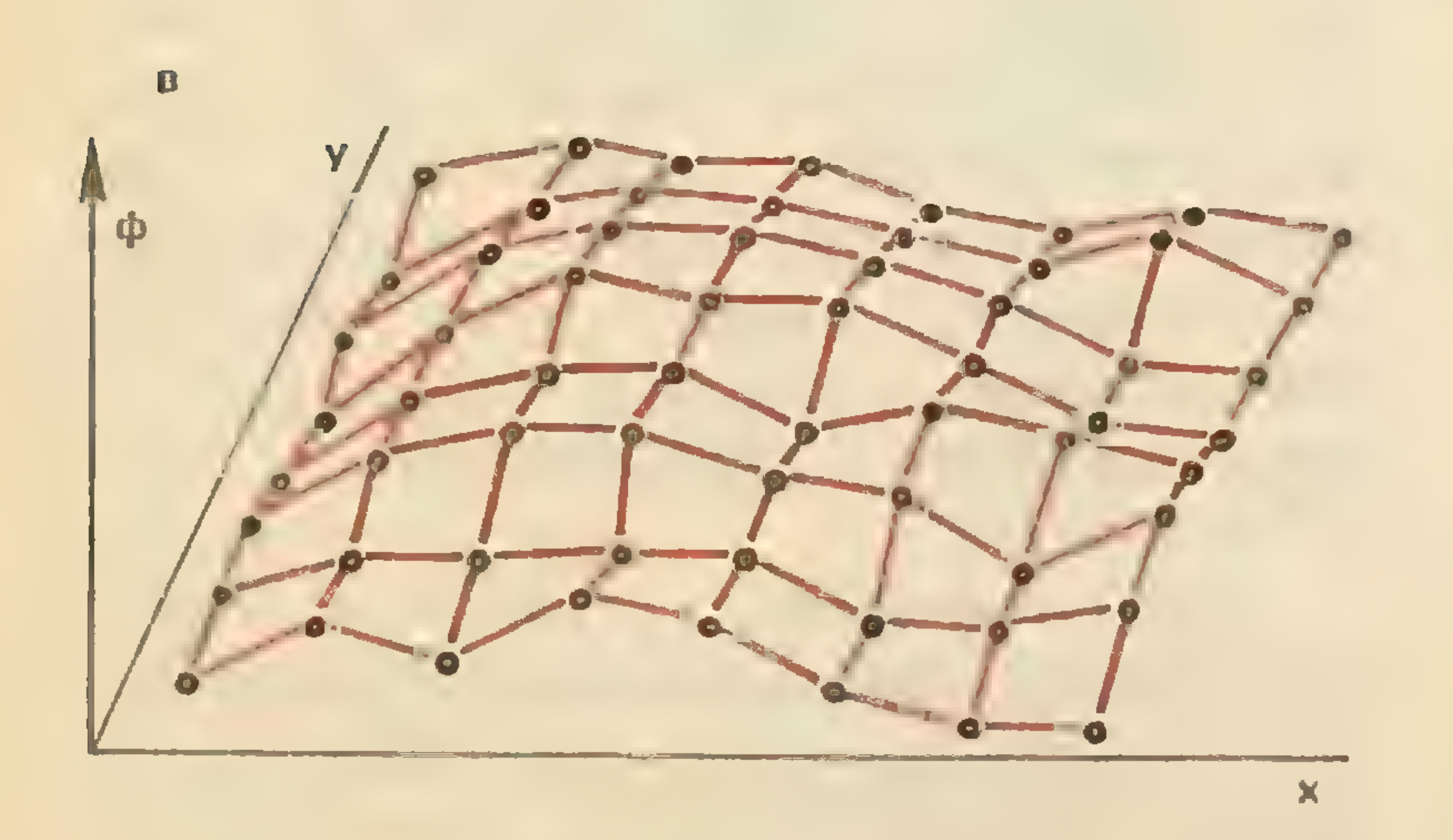
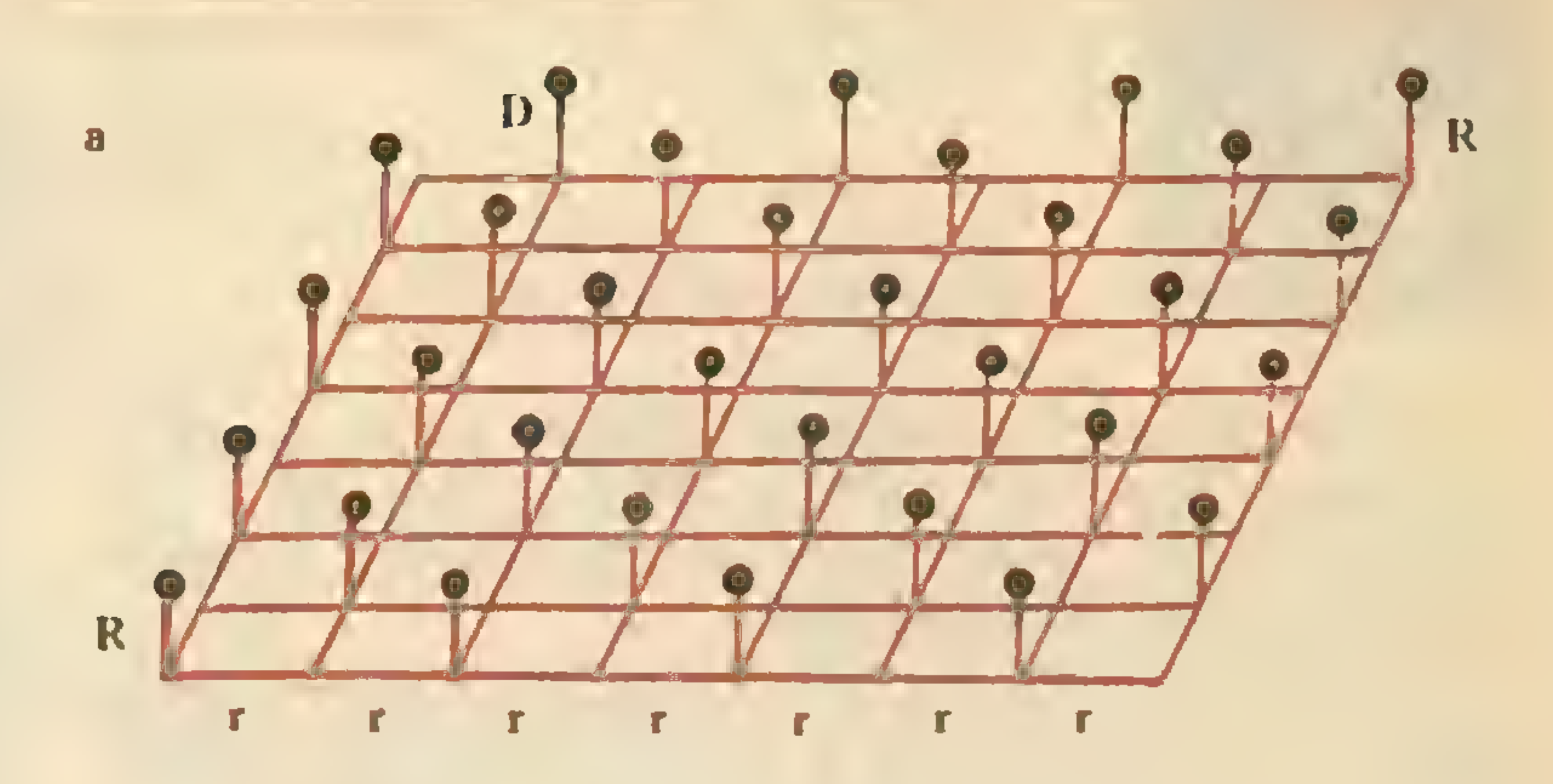
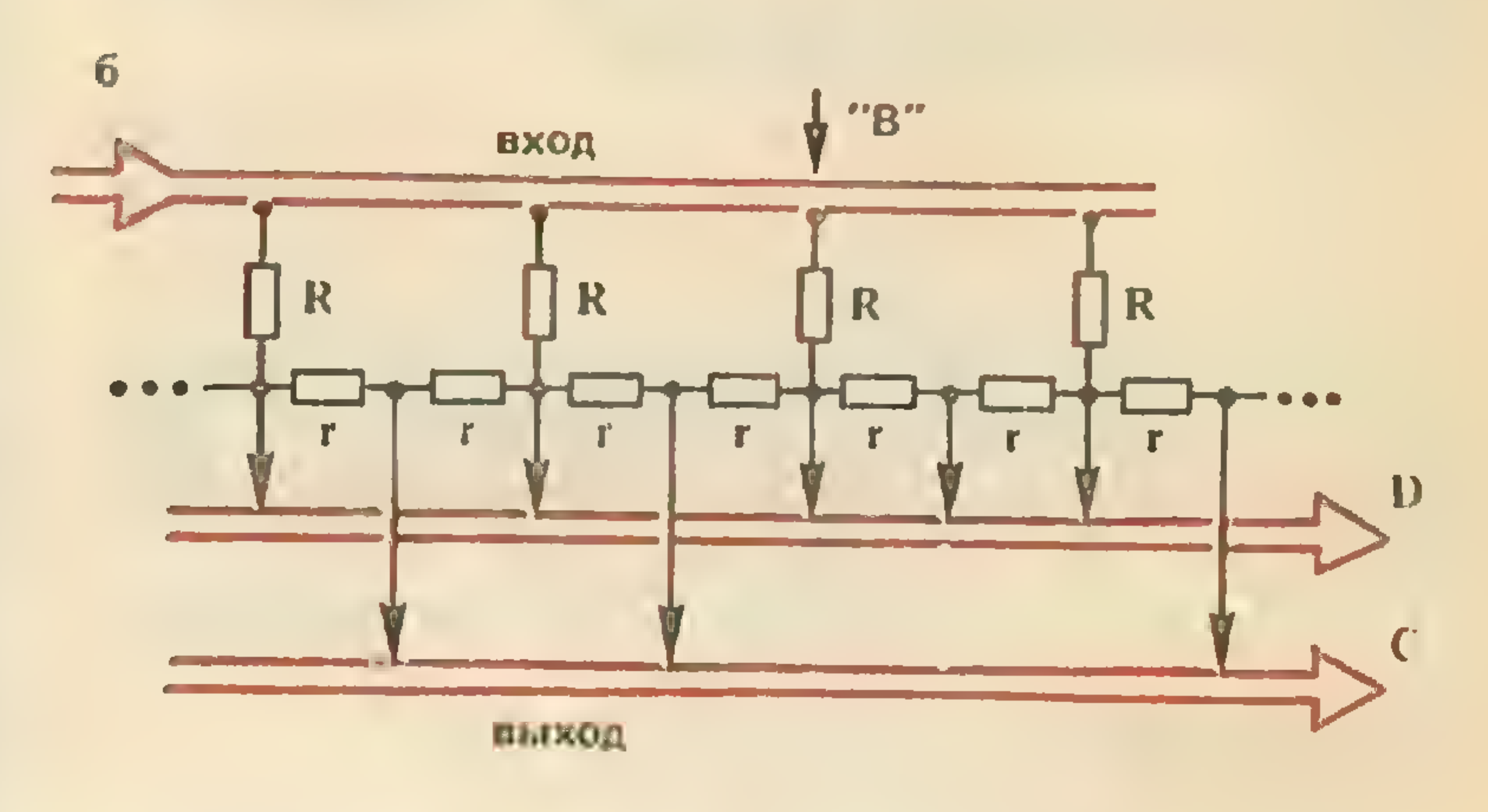


Рис. 2.





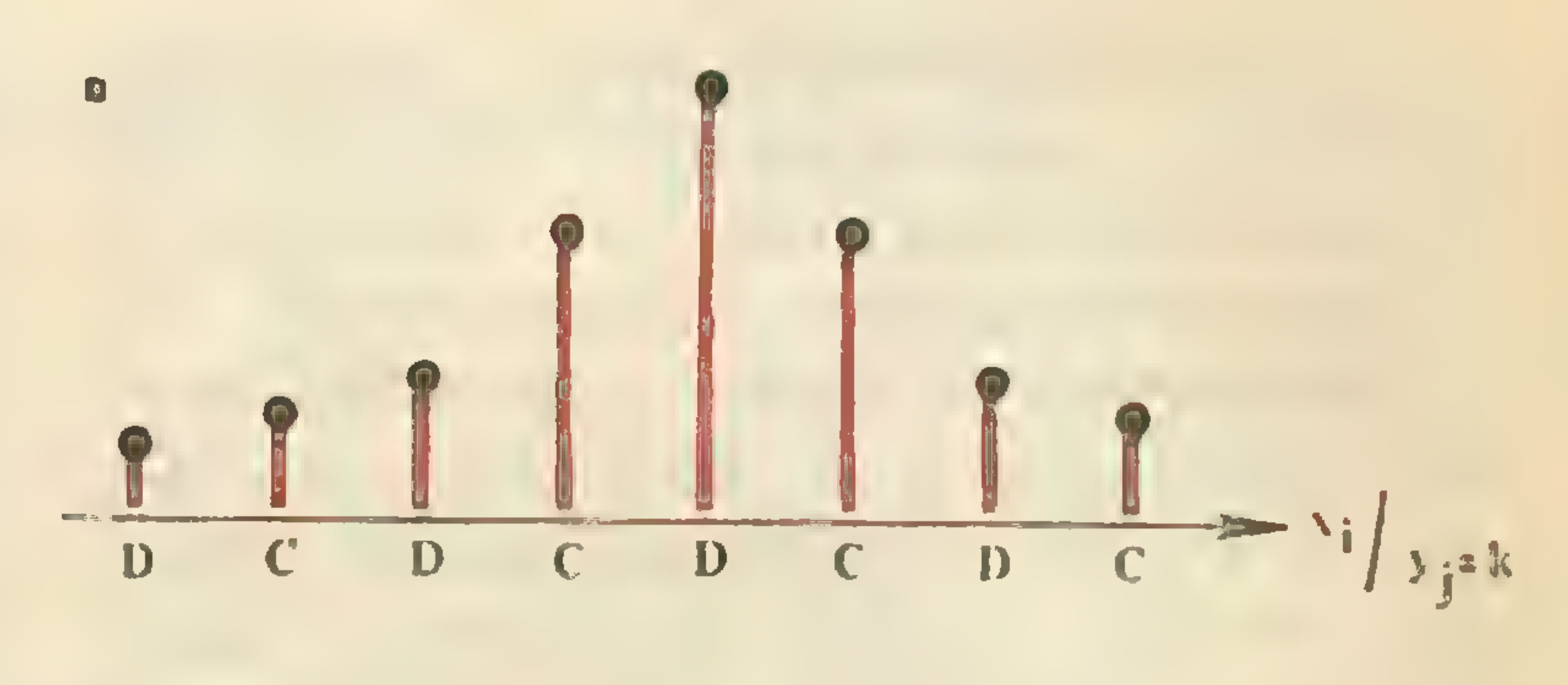


Рис. 3.

Вверху — общий вид системы «Кентавр». В центре расположен гибридный процессор с встроенными индикаторами, слева от него — осциллографический, а справа — телевизионный индикаторы. Внизу — образец комбинированного телевизионного изображения с яркостным распределением плотности нейтронов и цифровыми данными

сигналов о нем непосредственно в момент измерения производят считывание сигналов с выходов полей-преобразователей с помощью электронных коммутаторов. Осциллографическое и телевизионное изображения сформированы таким образом, чтобы информационной единицей было поле, соотнесенное с планом активной зоны. Это позволяет четко контролировать ход эксперимента, легко и просто прослеживать всю последовательность преобразований сигналов, отличать отказы системы от физических процессов в реакторе, обеспечивает технологичность и простоту настройки системы, упрощает диагностику в процессе эксплуатации.

Наряду с развитыми аналоговыми средствами преобразования и отображения сигналов имеется блок, осуществляющий отображение полученных данных в виде картограммы на дисплее, вывод картограмм на печать и ввод данных в ЭВМ «Мера-60». На ЭВМ производится обработка результатов в ходе эксперимента и после его завершения. Метрологические характеристики системы изучаются в процессе ее эксплуатации.

«Кентавр» был разработан и изготовлен силами студенческого конструкторско-исследовательского бюро по автоматике СКИБ-А МИФИ и введен в эксплуатацию на критическом стенде ИАЭ им. И. В. Курчатова менее чем за год и находится в эксплуатации уже более двух лет. За это время он зарекомендовал себя как удобный и надежный инструмент исследований. Экономический эффект превышает 150 тыс. руб. в год.

«Кентавр» развивает принципы индикатора физических полей «Полескоп», успешно работающего на Курской АЭС с 1982 г. В 1983 г. создание и внедрение индикатора «Полескоп» было отмечено дипломом Почета и восемью медалями ВДНХ СССР.

Более подробную информацию о «Полескопе» и «Кентавре» можно найти в литературе:

1. Алексаков Г. Н. и др. Индикатор энергораспределения поля РБМК-1000// — Атомная энергия. — 1982. — Т. 53. — Вып. 4. — С. 263—265.

2. Алексаков Г. Н. и др. Индикатор физических полей «Полескоп»// Приборы и техника эксперимента. — 1983. — № 1. — С. 215—216.

3. Алексаков Г. Н. и др. Опыт эксплуатации и перспективы использования системы контроля физических параметров «Полескоп» в составе СУЗ реакторов типа РБМК. — В кн.: Вопросы атомной науки и техники, сер. Физика и техника ядерных реакторов. — Вып. 3. — М., 1985.

4. Алексаков Г. Н. Федоров В. А. Корреляционно-диагностический контроль СУЗ РБМК. — В кн.: Вопросы атомной науки и техники, сер. Физика и техника ядерных реакторов. — Вып. 10. — М., 1985.

УВАЖАЕМЫЕ ПОДПИСЧИКИ!

В вып. 5 серии в статье Е. А. Соколова допущены опечатки. Список опечаток будет опубликован в одном из номеров этого года.

В 93 **В океане** данных. — М.: Знание, 1988. — 48 с. — (Новое в жизни, науке, технике. Сер. «Вычислительная техника и ее применение»; № 10).
15 к.

В брошюре рассказано о развитии средств и методов обработки данных в информационных системах, приведших в конце 60-х годов к появлению концепции баз данных. Рассмотрены СУБД, позволяющие достаточно просто выполнять основные манипуляции над данными: ввод информации в БД, эффективный поиск необходимой информации, формирование отчетов по содержимому БД в виде таблиц, графиков и т. п. Ознакомление с принципами и методами использования таких систем поможет специалистам в самых разных областях экономики, науки и техники оценить их возможности и найти им применение в своей работе.

Рассчитана на широкий круг читателей.

1602120000 **ББК 22.19**



Научно-популярное издание

В ОКЕАНЕ ДАННЫХ

Гл. отраслевой редактор *Л. А. Ерлыкин*
Редактор *Б. М. Васильев*
Мл. редактор *Н. А. Васильева*
Художники *В. Н. Конюхов* и *К. Н. Мошкин*
Худож. редактор *М. А. Гусева*
Техн. редактор *Т. В. Луговская*
Корректор *В. И. Гуляева*

ИБ № 9619

Сдано в набор 23.06.88. Подписано к печати 19.08.88. Т08463. Формат бумаги 70×100[1]/16. Бумага офсетная. Гарнитура журнально-рубленая. Печать офсетная. Усл. печ. л. 3,90. Усл. кр.-отт. 8,45. Уч.-изд. л. 4,26. Тираж 66 100 экз. Заказ 2338. Цена 15 коп. Издательство «Знание». 101835, ГСП, Москва, Центр, проезд Серова, д. 4. Индекс заказа 884710. Ордена Трудового Красного Знамени Калининский полиграфический комбинат Союзполиграфпрома при Государственном комитете СССР по делам издательств, полиграфии и книжной торговли. 170024, г. Калинин, пр. Ленина, 5.

ГОТОВЯТСЯ К ПЕЧАТИ В 1989 г.

Серия «Вычислительная техника и ее применение»

Персональные ЭВМ
Операционные системы
Микропроцессоры
Экспертные системы
Дружественное программное обеспечение
ЭВМ в искусстве
Машинная графика
Искусственный интеллект
Периферийное оборудование ЭВМ
Речь и ЭВМ
Технология информатики
Пакеты прикладных программ

Серия «Радиоэлектроника и связь»

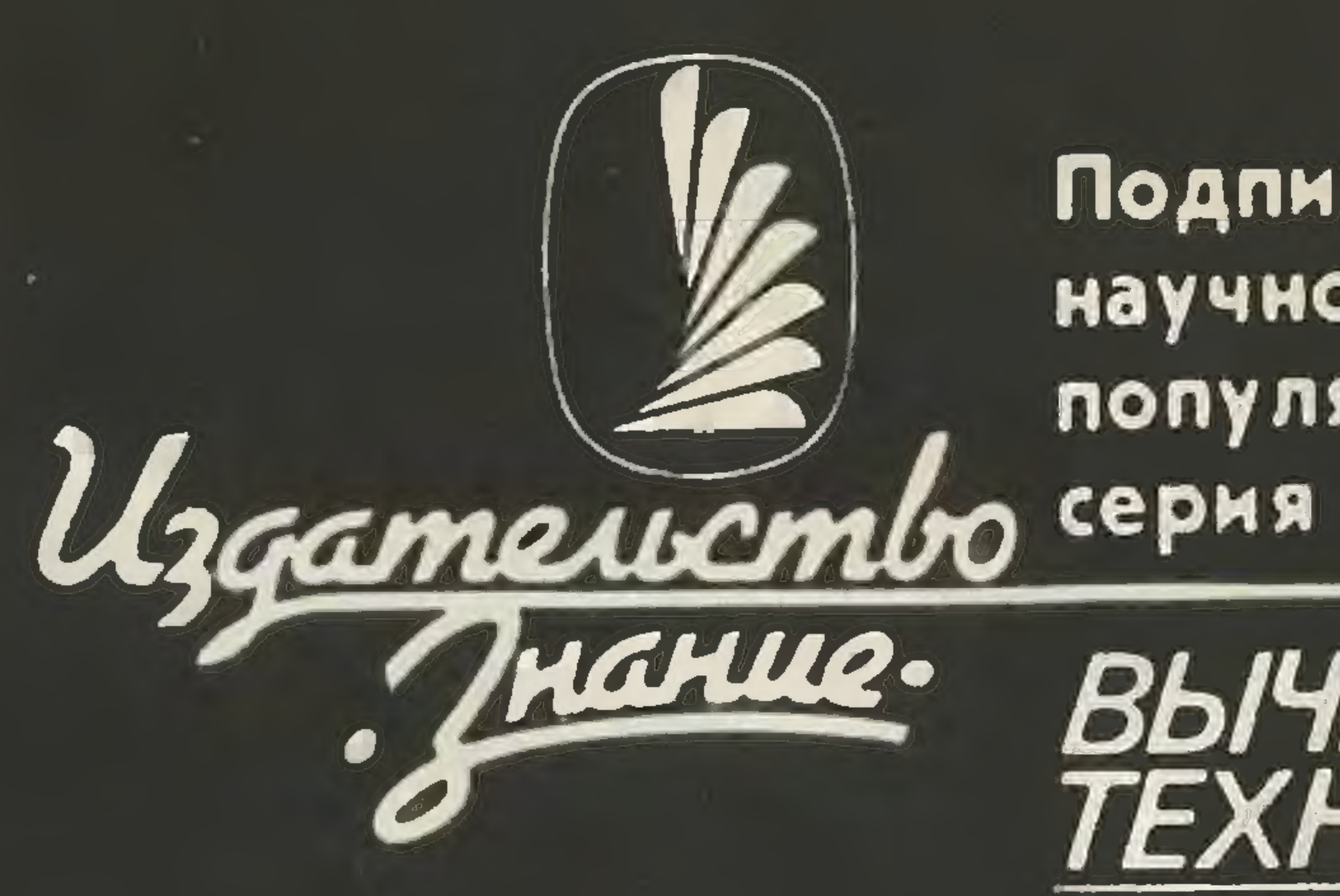
ЭВМ в системах связи
Цифровое телевидение
Кремний или арсенид галлия?
Электроника и экология
Нанотехнология
Компьютерный фазовый микроскоп — прибор технологии XXI века
Системы высококачественного воспроизведения фонограмм
Аппаратное обеспечение проектов «Вега» и «Фобос»
Радиоэлектронная борьба
Сверхрешетки (двумерный электронный газ)
Приборы с зарядовой связью
Нетрадиционные устройства машинной памяти

Цена 15 коп.

Индекс 70195

Адрес подписчика:

Лен. 27-43



Подписная научно-популярная серия

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

И ЕЁ ПРИМЕНЕНИЕ

Дорогой читатель!

Брошюры этой серии в розничную продажу не поступают, поэтому своевременно оформляйте подписку.
Подписка на брошюры издательства «Знание» ежеквартальная, принимается в любом отделении «Союзпечати».

Наш адрес: Москва, Центр, проезд Серова, д. 4